VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Rozpoznání textu v mobilním telefonu s OS Android

Recognize Text in the Mobile Phone with OS Android

2013 **Libor Stebel**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Libor Stebel**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: Rozpoznání textu v mobilním telefonu s OS Android
Recognize Text in the Mobile Phone with OS Android

Zásady pro vypracování:

1. Popis OS Android
2. Popis vývoje aplikace pro OS Android
3. Popis principu OCR a knihoven OCR pro programovací jazyk Java
4. Vytvoření aplikace rozpoznání textu v mobilním telefonu s OS Android

Seznam doporučené odborné literatury:

MURPHY, Mark L. Android 2: Průvodce programováním mobilních aplikací. Česká republika: COMPUTER PRESS, 2011. ISBN 9788025131947.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Marcel Fajkus**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

prof. RNDr. Vladimír Vašinek, CSc.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: 29. 4. 2013

.....
podpis studenta

Poděkování

Rád bych poděkoval Ing. Marcelovi Fajkusovi za odbornou pomoc a konzultaci při vytváření této bakalářské práce.

Abstrakt

Smyslem této bakalářské práce je vytvořit aplikaci v programovacím jazyce Java na platformě Android od společnosti Google. Cílem aplikace je rozpoznat text z obrázku nebo fotografie a následně tento text sdílet dále např. pomocí emailu, internetu atd. Aplikace by měla být schopná pořídit obrázek nebo fotografii buď ze souboru, nebo vyfocením z integrované kamery mobilního telefonu. Z načteného obrázku nebo fotografie si uživatel vybere oblast, kterou chce vyhodnotit. Aplikace provede skenování vybrané oblasti, v němž se provedou různé druhy filtrací pro zlepšení kvality obrázku, a následně porovnává nalezená písmena s databází vzorových písmen. Tento naskenovaný text může uživatel uložit do paměti telefonu, hledat na internetu pomocí webového prohlížeče, poslat emailem nebo přeložit do jiného jazyka. Nakonec této práce se aplikace otestuje pro různé druhy a velikosti písma.

Klíčová slova

Android, Java, Google, Optické rozpoznávání textu, Inteligentní rozpoznávání textu

Abstract

The purpose of this bachelory work is to create an application in programming language Java on the Android platform from Google Company. The aim of the application is to recognize text from a picture or photo and then share this text for example via email, Internet, etc. The application should be able to take picture or photograph either from a file or via the camera integrated in the mobile phone. The user selects the area from the loaded image or photo, which want to evaluate. The application will scan the selected areas with different types of filtration to improve the quality of image and then compares the found letters with database of comparative letters loaded before. The user can this scanned text store in phone memory, search on the Internet via web browser, send by email or translate to other language. Finally, this application will be tested for different types and sizes of fonts.

Key words

Android, Java, Google, Optical character recognition, Intelligent Character Recognition

Seznam použitých zkratek

Zkratka	Anglický význam	Český význam
A2DP	Advanced Audio Distribution Profile	Zlepšený profil distribuce audia
ADT	Android Development Tool	Nástroj pro vývoj Androidu
API	Application Programming Interface	Rozhraní pro programování aplikací
APK	Application package	Aplikační balíček
AVD	Android Virtual Device	Virtuální zařízení Androidu
CDMA	Code Division Multiple Access	Metoda digitálního multiplexování
DPI	Dots per inch	Počet obrazových bodů na palec
GPS	Global Positioning System	Globální polohový systém
HDR	High Dynamic Range	Vysoký dynamický rozsah
HTML5	HyperText Markup Language 5	Značkovací jazyk pro hypertext
HW	Hardware	Hardware
ICR	Intelligent Character Recognition	Inteligentní rozpoznávání znaku
IDE	Integrated Development Environment	Vývojové prostředí
JAR	Java archive	Balíček jazyku Java
JIT	Just-in-time	Metoda pro zrychlení kompilátoru
NFC	Near Field Communication	Bezdrátová technologie pro krátkou vzdálenost
NDK	Native Development Kit	Vývojový kit pro nativní kódy
OCR	Optical Character Recognition	Optické rozpoznávání znaku
OHA	Open Handset Alliance	Společnosti, které stojí za vývojem Androidu
OMR	Optical Mark Recognition	Optické rozpoznávání značek
OpenGL	Open Graphics Library	Knihovny pro práci s grafikou
OS	Operating System	Operační systém
QWERTY	Type of keyboard	Typ klávesnice
RAM	Random Access Memory	Paměť s náhodným přístupem
ROM	Read Only Memory	Paměť pouze pro čtení

SD	Secure Digital	Typ paměťové karty
SDK	Software Development Kit	Sada vývojových programů
SIP	Session Initiation Protocol	Protokol pro přenos signalizace v internetové telefonii
SMS	Short Message Service	Služba krátkých textových zpráv
SQLite	Database managment system	Databázový systém
SW	Software	Software
USB	Universal Serial Bus	Univerzální sériová sběrnice
VPN	Virtual Private Network	Virtuální privátní síť
Wi-Fi	Wireless Fidelity	Bezdrátová komunikace v počítačových sítích
XML	Extensible Markup Language	Rozšířitelný značkovací jazyk

Obsah

1 Úvod 1

2 Android 2

2.1 Historie 2

2.2 Architektura 2

2.2.1 Aplikace 2

2.2.2 Aplikační Framework 2

2.2.3 Knihovny 3

2.2.4 Android Runtime 3

2.2.5 Linuxové jádro 3

2.3 Verze 3

2.3.1 1.5 Cupcake 4

2.3.2 1.6 Donut 4

2.3.3 2.0 - 2.1 Eclair 4

2.3.4 2.2 Froyo 5

2.3.5 2.3 Gingerbread 5

2.3.6 3.1 - 3.2 Honeycomb 5

2.3.7 4.0 IceCreamSandwich 5

2.3.8 4.1 - 4.2 JellyBean 5

2.4 Vývoj aplikace 6

2.4.1 Prostředky pro vývoj 6

2.4.2 Složení aplikace 6

2.4.3 Složení projektu 6

2.4.4 Práva 7

2.4.5 Životní cyklus aktivit 7

2.4.6 Testování 9

2.5 Jednoduchá aplikace 9

2.5.1 Vytvoření nové aplikace 9

2.5.2 Vytvoření uživatelského rozhraní (pomocí tzv. layoutů) 9
2.5.3 Vytvoření aktivit 9
3 OCR 11
3.1 Historie OCR 11
3.2 OCR dnes 12
3.3 ICR - Intelligent Character Recognition 13
3.4 OMR - Optical Mark Recognition 13
3.5 Fáze převodu 14
3.5.1 Pořízení obrázku 14
3.5.2 Předzpracování obrázku 14
3.5.3 Převod 15
3.5.4 Korekce textu 15
3.6 Metody pro zlepšení vstupního obrazu 15
3.6.1 Převod obrázku na odstíny šedé 15
3.6.2 Geometrické transformace 15
3.6.3 Jasové transformace 16
3.6.4 Vyhlazování obrazu 16
3.6.5 Detekce hran 16
3.6.6 Prahování 17
3.7 Metody OCR 17
3.7.1 Střední kvadratická chyba MSE (Mean Squared Error) 17
3.7.2 Obrazové invarianty 17
3.8 Vliv typu písma na převod 18
4 Knihovny pro OCR v programovacím jazyce Java 19
4.1 Převody v cloudu 19
4.2 Převody na zařízeních 19
4.3 Knihovny 19
4.3.1 Abby FineReader Engine 19
4.3.2 Abby Mobile OCR Engine 4.0 19

4.3.3 Abby Cloud OCR SDK 20
4.3.4 Leadtools OCR SDK 20
4.3.5 Digital Syphon Sonic Imagën 20
4.3.6 Asprise OCR SDK v4.0 20
4.3.7 Saaspose.OCR SDK for Java 21
4.3.8 Aspose.OCR for Java 21
4.3.9 Tesseract - ocr 22
4.3.10 Java OCR 22
5 Mobilní aplikace 24
5.1 Softwarové vybavení 24
5.2 Hardwarové vybavení 24
5.3 Požadavky pro běh 24
5.4 Seznámení s aplikací 24
5.5 Popis jednotlivých aktivit a tříd 27
5.5.1 Aktivita MainActivity 27
5.5.2 Aktivita MyCamera 27
5.5.3 Aktivita EditImageSize 28
5.5.4 Aktivita EditScannedText 28
5.6 Použitá OCR knihovna 29
5.6.1 Předzpracování obrázku 29
5.6.2 Převod 29
5.6.3 Training data 30
5.6.4 Formáty vstupních obrázků 30
5.7 Testování 31
5.8 Zhodnocení 31
6 Závěr 33
Použitá literatura 34
Seznam příloh 35

1 Úvod

V dnešním světě se chytré telefony stávají čím dál více součástí našeho života. Od dob prvního mobilního telefonu prošly výraznou změnou. Dnes je využíváme pro mnoho dalších funkcí oproti telefonování a posílání SMS. Používáme je pro pořizování fotografií, natáčení videa, prohlížení internetu, posílání emailů, prohlížení dokumentů a mnoho dalších funkcí vytvořených programátory. Za dobu trvání chytrých telefonů vzniklo také mnoho operačních systémů od různých společností. Většina z nich se však neudržela na dnešním náročném trhu a přestala se vyvíjet. Na vrcholu je dnes operační systém Android od společnosti Google, který si nejvíce konkuruje s iOS od společnosti Apple.

První část této práce se zabývá právě Androidem, který se stal populární svou hardwarovou přenositelností, programátorskou přívětivostí a dostupností pro jednotlivé výrobce mobilních zařízení. V této části se dozvíte něco o historii, architektuře, o jednotlivých verzích a vývoji aplikace. Nakonec je zde příklad jednoduché aplikace.

Dnes existují softwarové nástroje zabývající se digitalizací tištěného textu, který by se jinak musel přepisovat ručně. V druhé části zjistíte něco o historii a principu technologie OCR, která vychází vstříc těmto dnešním požadavkům na digitalizaci tištěných textů. Tato část dále popisuje různé metody používané v této technologii. Následují knihovny dostupné pro OCR v programovacím jazyce Java.

Poslední část se zabývá mobilní aplikaci pro Android, která implementuje předchozí témata. Jsou zde popsány jednotlivé kroky a třídy aplikace, dále problémy se kterými se lze setkat a testování různých vstupních obrázků. Celá práce je zakončena zhodnocením, ve kterém je uvedena kvalita aplikace a návrhy na možné vylepšení.

2 Android

Android je open source platforma od společnosti Google vytvořena zejména pro mobilní zařízení s menší velikostí paměti, výdrží baterie a výkonnosti, jako jsou smartphony, navigace a tablety. Je složena z operačního systému, založeném na linuxovém jádře, které bylo navrženo pro běh na různém hardwaru, dále z middleware, uživatelského rozhraní a aplikací. Kromě toho využívá svůj vlastní virtuální stroj pro optimalizaci paměti a hardwarových zdrojů v mobilním prostředí.[2] Dnes běží Android na stovkách miliónů mobilních zařízení a denně se přidává milion dalších.[3]

2.1 Historie

OS Android začala vyvíjet firma Android. V srpnu 2005 byla společnost Android koupena společností Google, která oficiálně 5. listopadu 2007 představila ukázkovou verzi SDK Androidu. Tato verze obsahovala vývojové prostředí, emulátor, debugger a ukázkové programy. Zároveň bylo založeno konsorcium Open HandsetAlliance (dále OHA), pod které spadalo celkem 34 zakládajících členů, a postupně se přidávaly další společnosti. Cílem OHA je vytvářet a prosazovat standardy vývoje SW a HW pro mobilní zařízení. Dnes již téměř všichni výrobci mobilních telefonů spolupracují s OHA (výjimkou jsou Nokia a Apple).[4]

Vůbec první verze OS Android 1.0 byla představena 23. září 2008 a poslána do rukou vývojářů. Zásadní událostí ale bylo představení mobilního telefonu T-Mobile G1 (HTC Dream). Tento telefon jako první dokázal prorazit s Androidem do světa a to také díky GPS navigaci, 3.1 Mpx fotoaparátu a hardwarovou QWERTY klávesnici. Od 22. října byl veřejnosti zpřístupněn Android Market, který tehdy obsahoval něco přes 30 aplikací.

Jedna z dalších veledůležitých věcí se udála v říjnu roku 2008. Android byl uvolněn jako open-source a jeho kompletní zdrojový kód mohl využívat každý. Snad i proto po něm sáhlo takové množství výrobců, kteří svou masivní produkcí modelů s OS od Googlu pomohli systému k současným výsledkům.[5]

2.2 Architektura

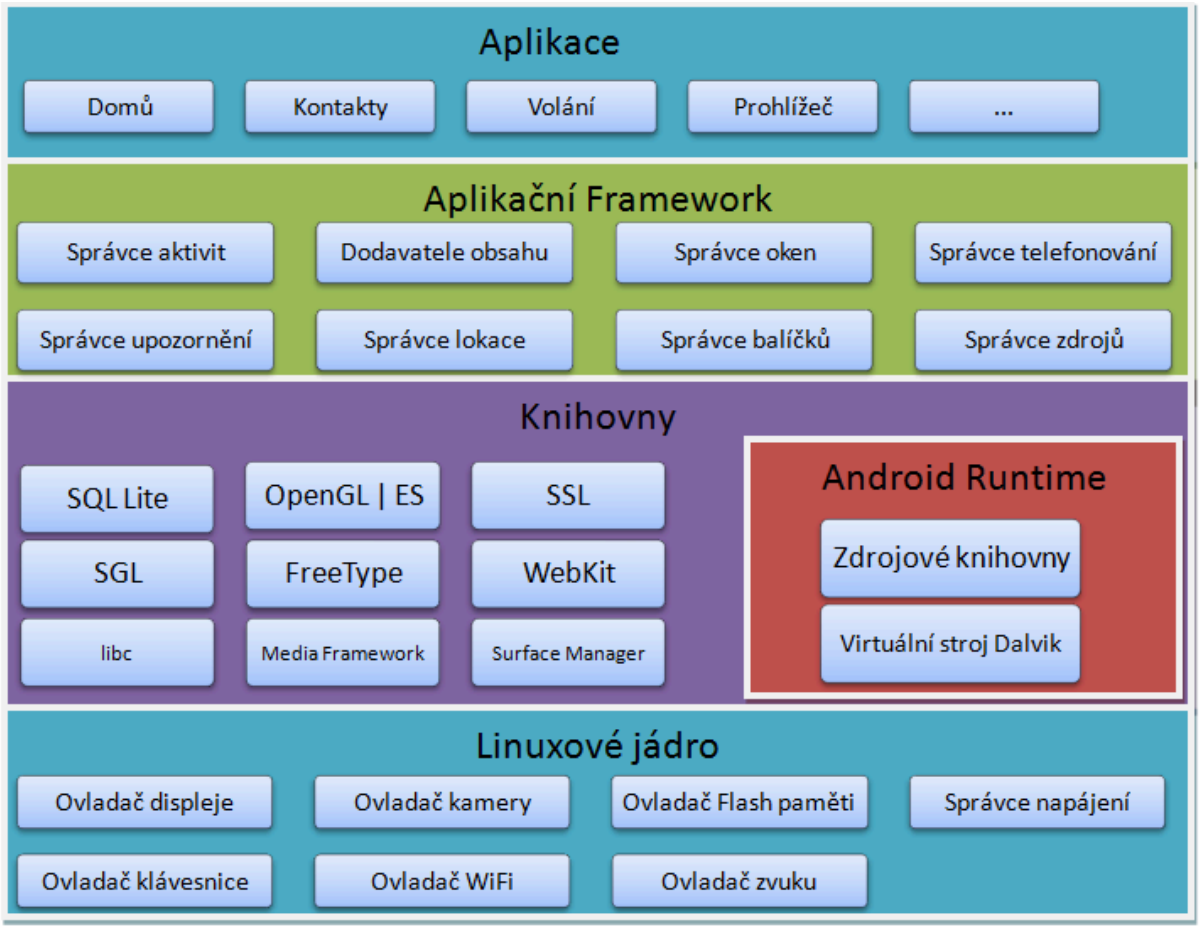
Skládá se z pěti vrstev, které jsou odděleny svoji funkčností. Celkově je vše navrženo a optimalizováno pro méně výkonný hardware. Architektura je znázorněna na obrázku 2.1.

2.2.1 Aplikace

Do této vrstvy patří veškeré aplikace naprogramované v jazyce Java. V základu už Android nabízí několik základních předinstalovaných aplikací, jako jsou kontakty, kalendář, email, zprávy, mapy, webový prohlížeč a další.

2.2.2 Aplikační Framework

Tato vrstva slouží vývojářům pro přístup k jednotlivým knihovnám, které ovládají různé komponenty zařízení. To jim poskytuje využívat veškeré schopnosti zařízení naplno, kupříkladu přístup k lokačním informacím, spouštění služeb na pozadí, přidávání notifikací do stavové lišty a mnoho dalších možností.



Obrázek 2.1: Architektura Androidu [6]

2.2.3 Knihovny

Jednotlivé knihovny napsány v jazyce C/C++ ovládající různé komponenty zařízení. Tyto knihovny už vývojář není schopen upravovat pro své účely. Patří sem např. knihovny Media Library, System C Library, SQLite, OpenGL, FreeType a další.

2.2.4 Android Runtime

V této části Androidu se stará o chod aplikací virtuální stroj nazvaný Dalvik. Dalvik je navržen tak, aby mohlo více aplikací běžet současně. Všechny soubory jsou zkompilovány do Dalvikovského formátu optimalizovaného pro minimální vytížení paměti.

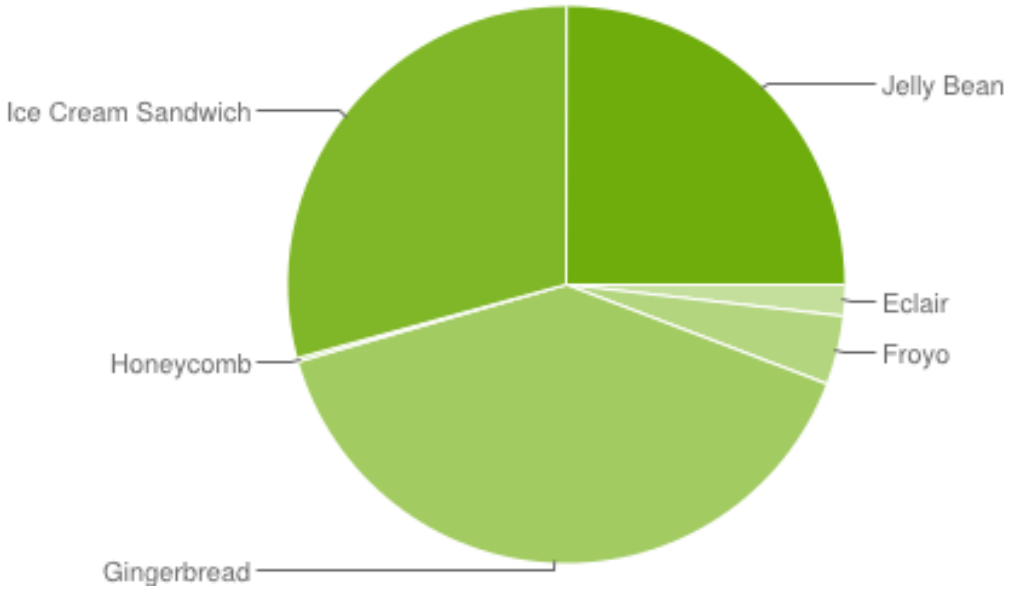
2.2.5 Linuxové jádro

Tato nejnižší vrstva působí jako přechod mezi hardwarovou a softwarovou strukturou. Stará se o hlavní služby systému jako je bezpečnost, správa paměti, procesů, sítě a jednotlivých ovladačů.[6]

2.3 Verze

Od počátku se Android neustále vyvíjí, zlepšuje a ladí. Postupem času vycházejí nové verze, které buď opravují chyby na starších verzích, nebo přidávají nové funkční prvky spojené často

s novým výkonnějším hardwarem. Současná verze Androidu (k datu 13. 1. 2013) je 4.2 a nese označení JellyBean. Na obrázku 2.2 je znázorněno aktuální zastoupení jednotlivých verzí.



Obrázek 2.2: Zastoupení verzí Androidu (k datu 2. 4. 2013) [7]

2.3.1 **1.5 Cupcake**

Dne 30. dubna 2009 se objevila první výraznější aktualizace. Ta přinesla lepší softwarovou klávesnici se slovníkem pro vlastní slova, nové widgety a složky, nahrávání a pouštění videa, podporu Bluetooth standartu A2DP a automatické párování zařízení, vylepšený webový prohlížeč s funkcemi kopírovat, vložit a hledání, vylepšení kontaktů s možností přiřazení obrázku k oblíbenému kontaktu, nahrávání videí na Youtube a fotek na Picasa přímo z telefonu a animované přechody mezi obrazovkami. Co se týče systému, bylo vylepšeno linuxové jádro (upgrade na 2.6.27) a operace s paměťovou kartou.

2.3.2 **1.6 Donut**

V září roku 2009 byla uvolněna verze 1.6. Ta s sebou přinesla nové vyhledávání v různých zdrojích přímo z domovské obrazovky, nové prostředí a větší rychlost fotoaparátu, kamery a galerie, podporu technologií VPN a CDMA, ukazatel vytíženosti baterie, zlepšené vyhledávání hlasem, podporu pro další rozlišení displeje a upgrade jádra na 2.6.29. Zlepšení se dostavilo také na Android Market.

2.3.3 **2.0 - 2.1 Eclair**

V říjnu 2009 přišla jedna z dalších velkých aktualizací a to na verzi 2.0. Tato aktualizace podstatně zrychlila běh celého systému. Nově se objevili víceprvkové kontakty s možností synchronizace s internetem, podpora Microsoft Exchange, další funkce fotoaparátu včetně digitálního zoomu a přisvětlovací diody, podpora vyšších rozlišení a animované tapety na domovské stránce. Vylepšení doznali virtuální klávesnice, webový prohlížeč nově s podporou HTML5, Google mapy 3.12 a Bluetooth 2.1.

2.3.4 2.2 Froyo

V květnu 2010 přišla aktualizace, která opět zrychlila běh celého systému díky JIT (Just-in-time) kompilátoru. Nově se zde objevila možnost instalování aplikací na paměťovou kartu, vytvořit z telefonu Wi-Fi hotspot nebo sdílet internetové připojení pomocí USB kabelu. Dále přibylo API pro práci s OpenGL ES 2.0, možnost obnovení do továrního nastavení a zaheslování telefonu. Vylepšen byl opět fotoaparát a kamera, Microsoft Exchange, Bluetooth a trackball podporující více barev. Jádro se upgradovalo na 2.6.32, což přineslo podporu RAM větších než 256MB.

2.3.5 2.3 Gingerbread

Zatím nejrozšířenější verze byla představena v prosinci 2010. Uživatelské prostředí se změnilo za účelem lehčího naučení a rychlejšího a výkonnějšího použití. Nově se zde objevila podpora SIP protokolu pro internetové volání, NFC standart, podpora video formátu WebM pro HTML5 video a na marketu se objevily Google Maps5 s 3D přístupem. Vylepšená byla klávesnice, funkce kopírovat a vložit a správce aplikací. Telefon nově podporoval více fotoaparátů a senzorů. Co se týče systému, tak byl vylepšen Dalvik a jádro upgradovalo na 2.6.35.

2.3.6 3.1 - 3.2 Honeycomb

V únoru 2011 byla představena první verze Androidu určená pro zařízení s velkými displeji, tedy pro tablety. Celé uživatelské rozhraní bylo předěláno pro potřeby tabletů. Jako první se zde objevila podpora pro vícejádrové procesory, vylepšený byl i multitasking a grafické operace. Přibyl USB port pro připojení např. klávesnice, u Google Talk se poprvé objevují videohovory a lze přistupovat ke Google eBooks. Z programátorského hlediska je nové také rozčlenění Aktivit (popsáno v kapitole 2.4.2) na tzv. Fragmenty.

2.3.7 4.0 IceCreamSandwich

V říjnu 2011 vyšla verze, která měla být už stejná pro telefony i tablety. Nabízí propracovanější multitasking, pohybové a dotykové gesta, možnosti při zamčené obrazovce, rozpoznávání hlasu, sdílení dat, galerie, prohlížeč, email, fotoaparát s funkcemi jako rozpoznání obličeje, panorama a další. Nově nabízí změnu velikosti widgetů (u tabletu už ve verzi 3.0), odpověď zprávou při příchozím hovoru, monitorování přenesených dat, screenshoty, Android Beam pro sdílení přes NFC, odemykání zařízení pomocí obličeje, Wi-Fi Direct a Bluetooth HDP. Linuxové jádro upgradovalo na 3.0.1.

2.3.8 4.1 - 4.2 JellyBean

V červenci 2012 se objevila zatím nejnovější verze Androidu, která nabízí opět zrychlení celého systému. Poprvé je zde možnost využít grafickou jednotku pro výpočty, což výrazně zvyšuje výkon zařízení. U tabletů nabízí přístup z jednoho zařízení k datům a aplikacím dalšího zařízení. Dále nabízí možnost přidat widget na zamčenou obrazovku, používat externí displeje bezdrátově, rozšířené mezinárodní volby a vylepšený fotoaparát s HDR režimem. Zdokonalení se týká i bezpečnostních prvků a vývojářských možností.[7]

2.4 Vývoj aplikace

Naučení této platformy zabere nejméně času ve srovnání s ostatními platformami (v průměru pouze 5 měsíců).[10] Avšak než začnete programovat aplikaci pro Android, musíte splnit několik podmínek, co se týče nástrojů pro vývoj.

2.4.1 Prostředky pro vývoj

Oficiálním podporovaným IDE je zde Eclipse. Lze použít i jiné nástroje, ale do nich nelze nainstalovat plugin ADT (Android Development Tools), který výrazně usnadňuje práci vývojářům. ADT umožňuje okamžitou kontrolu zdrojových kódů Javy a XML, debugování jak na skutečném tak i na virtuálním zařízení přímo v IDE, monitorování paměti i procesoru a další užitečné funkce. Navíc obsahuje editor pro vytváření uživatelských rozhraní. Další nezbytnou součástí je Android SDK, neboli balíček API knihoven a nástrojů pro vývoj, testování a debugování aplikací. Uživatel si pomocí SDK Manageru může stáhnout balíček pro jakoukoliv verzi Androidu popřípadě rozšířit již stávající balíček o doplňkové funkce. Poslední součást se už netýká přímo Androidu, ale obecně programovacího jazyku Java. Proto je nutné mít nainstalované nástroj pro práci s tímto jazykem nazvaný Java Development Kit.[8]

2.4.2 Složení aplikace

- **Activities** (Aktivity) - stavební kámen pro tvorbu uživatelského rozhraní. Můžeme je přirovnat k oknům nebo dialogům aplikace na stolní počítač. Aktivity nemusí mít žádné uživatelské rozhraní, avšak většina kódu bez uživatelského rozhraní bude zabalena spíše ve formě dodavatelů obsahu anebo služeb.
- **Content Provider** (Dodavatel obsahu) - dovoluje zpřístupnit data uložená v telefonu jiným aplikacím a současně poskytuje úplnou kontrolu nad způsobem přístupu k těmto datům.
- **Services** (Služby) - jsou prvky, které dokážou běžet nezávisle na jakékoliv aktivitě i po jejím ukončení. Například u přehrávání hudby pokračuje i po vypnutí příslušné aktivity.
- **Broadcast Reciever** (Záměr) - jsou systémové zprávy upozorňující aplikace na výskyt různých událostí. Tyto události můžou být např. vložení SD karty, příchozí SMS, nebo další námi vytvořené. Na tyto záměry můžeme reagovat a spouštět jejich prostřednictvím jiné aktivity.

2.4.3 Složení projektu

Obsahuje veškeré zdrojové kódy aplikace jak pro logiku, tak pro uživatelské rozhraní, veškeré multimediální soubory, Java archivy (JAR) třetích stran a další. Všechny tyto soubory se dohromady zabalí do souboru APK (Android package), který je pak podporován telefony a emulátory.

- **Android Manifest** - klíčová součást projektu, která obsahuje veškeré komponenty aplikace a jejich práva. Manifest používá Android za chodu aplikace k jejímu propojení s operačním systémem. Je využíván také na Android Marketu, kdy například nedovolí nainstalovat aplikaci, která využívá fotoaparát, telefonům bez fotoaparátu. Ukázku z manifestu můžete vidět na příkladu 2.1.

- **Assets/** - složka obsahující ostatní statické soubory, které chcete přibalit k aplikaci pro použití na zařízení.
- **Bin/** - složka, která uchovává aplikaci po jejím zkompilování.
- **Gen/** - složka, do které překladové nástroje systému Android umístí zdrojový kód, který vygenerují.
- **Libs/** - složka uchovávající Java archivy třetích stran.
- **Src/** - složka uchovávající zdrojový kód aplikace.
- **Res/** - složka uchovávající např. multimediální soubory, návrhy grafického rozhraní, texty apod. Na tyto soubory se pak odkazujeme ve zdrojovém kódu pomocí unikátních ID, které se vytvoří automaticky.

2.4.4 Práva

Deklarují se v manifestu. Ukázku můžete vidět na příkladu 2.1.

- **uses-permission** - práva, která bude naše aplikace potřebovat, aby mohla správně fungovat. Např. že k aplikaci je potřeba fotoaparát.
- **permission** - práva pro ostatní aplikace, aby mohli používat logiku nebo data z této aplikace.
- **instrumentation** - deklarují zdrojový kód, který se má spustit při určité systémové události.
- **uses-library** - připojují volitelné android komponenty.
- **uses-sdk** - udává verzi androidu pro kterou je aplikace přeložena.
- **application** - udává informace o aplikaci jako název, ikona atd. a jednotlivé aktivity ze kterých je aplikace složena.[1]

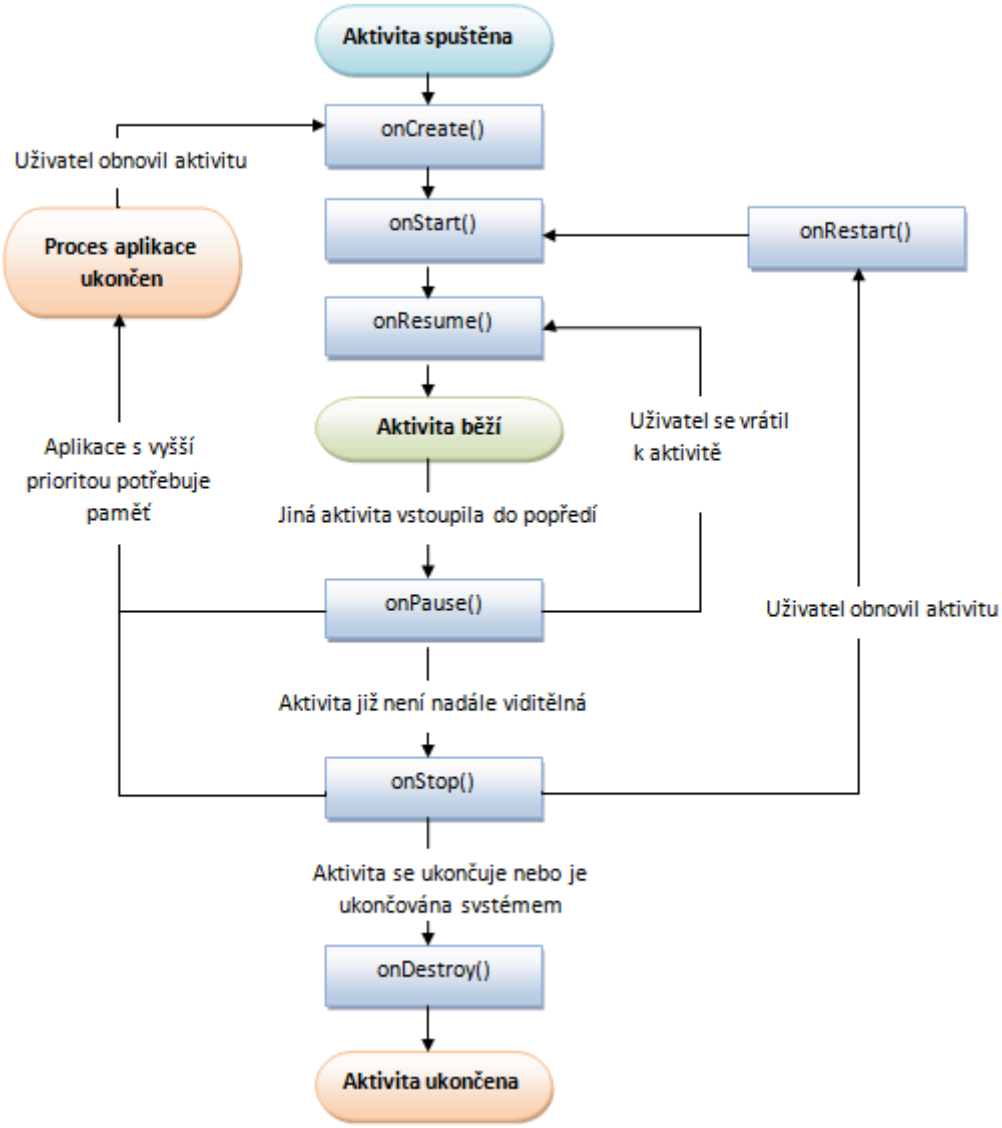
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.scaner" >
<uses-sdk
android:minSdkVersion="7"
android:targetSdkVersion="16" />
<uses-permission android:name="android.permission.CAMERA" />
<application
android:icon="@drawable/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme" >
...
</application>
</manifest>
```

Příklad 2.1: Ukázka manifestu.

2.4.5 Životní cyklus aktivit

Aktivita obsahuje několik metod, které se volají při určitých událostech jako např. příchozí hovor, zhasnutí displeje, první spuštění aktivity a mnoho dalších událostí. Životní cyklus si můžete prohlédnout na obrázku 2.3.

- **onCreate** - volá se při prvním vytvoření aktivity, zde by měly proběhnout všechny počáteční nastavení. Vždy následuje metoda `onStart`.
- **onRestart** - metoda se volá, pokud byla aktivita zastavena a chceme ji znova spustit.
- **onStart** - tato metoda je volána, když se aktivita stane viditelná pro uživatele.
- **onResume** - tato metoda se volá, když začne být aktivita schopna komunikovat s uživatelem.
- **onPause** - slouží pro zastavení aktuální činnosti při přepnutí do jiné aktivity. Implementace této metody by měla být rychlá, protože následující aktivita nezačne, dokud tahle metoda neskončí.
- **onStop** - volána, když aktivita není už dále viditelná pro uživatele.
- **onDestroy** - volána při ukončení aktivity.[9]



Obrázek 2.3: Životní cyklus aktivit [9]

2.4.6 Testování

V IDE Eclipsu máme možnost testování aplikace přímo na připojeném zařízení, nebo na tzv. emulátoru, který je součástí Android SDK a slouží pro testování aplikace virtuálně. U reálného zařízení je výhoda, že vidíme přesně, jak se aplikace chová na daném zařízení. Avšak dnes existuje velké množství zařízení, což by stálo nemalé prostředky pro jejich nákup. Na emulátoru lze nasimulovat téměř kterékoliv zařízení. Pro simulaci reálného zařízení si vytvoříme virtuální zařízení s OS Android tzv. AVD (Android Virtual Device), kterému můžeme nastavit verzi Androidu, hardwarové prvky, velikost displeje a další. Nevýhodou je však jeho rychlost a fakt, že některé data musíme také simulovat (např. Fotoaparát, GPS, senzory atd.).[1]

2.5 Jednoduchá aplikace

Pro vyzkoušení předešlých informací je zde nyní popsán vývoj jednoduché aplikace, které zadáme 2 čísla, a na další obrazovce nám vypíše jejich součet. Popisovaná bude především tvorba přes grafické editory, které nabízí ADT plugin.

2.5.1 Vytvoření nové aplikace

V IDE Eclipsu se vytvoří nový Android projekt, kde se zadá název aplikace, ostatní názvy se doplní automaticky. Dále se vybere, pro které verze androidu se bude aplikace vytvářet. V dalším okně se zaškrtne Create activity (vytvoření aktivity), které se na pozdějším okně musí pojmenovat. Nic jiného zatím není třeba nastavovat. Vytvoří se základní struktura adresářů a pár základních souborů.

2.5.2 Vytvoření uživatelského rozhraní (pomocí tzv. layoutů)

Ve složce `res/layout` se vytvořil soubor pro konfiguraci uživatelského rozhraní ve formátu XML. V základním tvaru je tam jedno textové pole s nápisem „Hello world!”. Hodnota tohoto textu je uložena v souboru `res/values/strings.xml`, kde se přepíše na např. „Zadejte dvě čísla k sečtení.” a uloží. Z nabídky `FormWidgets` se přidá obyčejným přetažením `Button` (tlačítko) a z `TextFields` se přidají dvě textové pole se vstupním typem Number. Text tlačítka je možno změnit v panelu `Properties` (nastavení) pod položkou Text. Nyní se ve složce `src/` a vytvořeném balíčku musí vytvořit nová Android aktivita s novým layoutem, který se nemusí měnit.

2.5.3 Vytvoření aktivit

Ve složce `src/` a uživatelsky pojmenovaném balíčku nyní jsou dvě aktivity, které už obě mají naimplementované základní tělo aplikace. Metodu `onCreateOptionsMenu` je teď nepotřebná, ta slouží pro vytvoření menu s nastavením aplikace. V těle třídy se nadefinují jednotlivé komponenty, které byly přidány do layoutu, importují se potřebné knihovny a v metodě `onCreate` se deklarují pomocí jejich jednoznačných ID, viz příklad 2.2. Nyní je potřeba reagovat na kliknutí tlačítka. Při vybraném tlačítku v panelu `Properties` pod položkou `onClick` se napíše název metody, která se bude volat při kliknutí na tlačítko, a pod stejným názvem se dopíše do třídy. Tato metoda však musí mít speciální syntaxi, viz příklad 2.2, kde je tato metoda nazvána „Spocitej”. V této metodě se získají hodnoty z jednotlivých textových polí a předají je další aktivitě pomocí tzv. Intentu.

```
public class MainActivity extends Activity {
      TextView nadpis;
      Button tlacitko;
      EditText cislo1,cislo2;

      @Override
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
            nadpis = (TextView)findViewById(R.id.textView2);
            tlacitko = (Button)findViewById(R.id.button1);
            cislo1 = (EditText) findViewById(R.id.editText1);
            cislo2 = (EditText) findViewById(R.id.editText2);
      }

      public void Spocitej(View v)
      {
            float num1 = Float.valueOf(cislo1.getText().toString());
            float num2 = Float.valueOf(cislo2.getText().toString());

            Intent intent = new Intent(this,Second.class);
            intent.putExtra("cislo1", num1);
            intent.putExtra("cislo2", num2);
            startActivity(intent);
      }}
```

Příklad 2.2: Ukázka aktivity

V následující vytvořené aktivitě se opět deklarují komponenty a v metodě `onCreate` se získají data z předchozí aktivity a vypíše jejich součet na displeji, viz příklad 2.3.

```
public class Second extends Activity {
      TextView nadpis;

      @Override
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_second);
            nadpis = (TextView)findViewById(R.id.textView);

            Bundle extras = getIntent().getExtras();
      if (extras == null) {
      return;
      }

            float cislo1 = extras.getFloat("cislo1");
            float cislo2 = extras.getFloat("cislo2");
      nadpis.setText(String.valueOf(cislo1+cislo2));
         }}
```

Příklad 2.3: Ukázka aktivity 2

3 OCR

OCR, z anglického Optical Character Recognition, česky optické rozpoznávání znaků, je technologie sloužící pro převod tištěných textů do elektronické podoby. Tištěný text je postupem času nahrazován právě textem digitálním, který vyniká přenositelností, jednoduchou úpravou, několikanásobným tištěním a podobně.

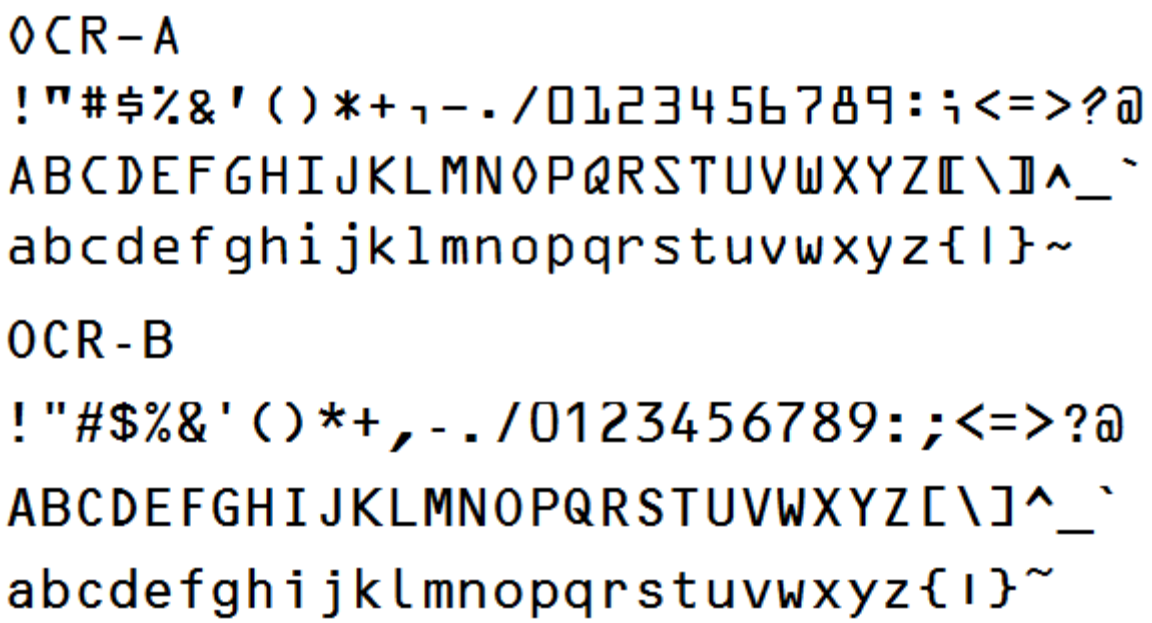
3.1 Historie OCR

Vůbec první náznaky OCR se objevily už v roce 1929, kdy rakouský vynálezce Gustav Tauschek vynalezl a následně si nechal patentovat tzv. „Čtecí stroj" (Reading Machine). Tento stroj fungoval na principu tzv. shody šablon, ze kterého dodnes vychází základní koncepty OCR. Pokud se daný znak překrýval se šablonou, stroj ho prohlásil za shodný. Patent odkoupila firma IBM a dodnes se u některých aplikací používá tento princip.

Dalším průkopníkem byl americký kryptoanalytik David H. Shepard, který v roce 1950 pracoval na automatizaci přepisování tištěného textu do upravovatelné elektronické formy. Tento systém si pak společně s jeho kamarádem Harveyem Cookem nechali patentovat pod názvem „Aparát na čtení" (Apparatus for reading). Po pár letech si opět Shepard tentokrát s Williamem Lawlessem založili firmu Intelligent Machines Research Corporation a vytvořili stroj zvaný „Gismo". Následně jejich patenty odkoupila firma IBM, která zdokonalovala systém rozpoznávání a poprvé použila pro tento systém název Optical Character Recognition. Tento název se používá dodnes, i když v dnešní době systém stojí na digitálním principu, nikoliv na opticko-mechanickém.

V 60. letech 20. století byly vytvořeny dva standardy písma, které byly vhodné pro systémy OCR. Jeden americký nazývaný OCR-A (viz obrázek 3.1) a druhý evropský nazývaný OCR-B (viz obrázek 3.1), který byl brán jako lépe čitelný pro lidi. První komerční OCR začaly používat pošty Spojených států amerických a Velké Británie.

V roce 1974 vznikl vývoj prvního počítačového OCR programu ve společnosti Kurzweil Computer Products, kterou založil Ray Kurzweil. Vývoj byl zaměřen hlavně pro zrakově postižené, kteří by lépe porozuměli tištěnému textu. Pro tento program museli vyvíjet i syntetizátor řeči pro počítače a 13. ledna 1976 úspěšně dokončili program nazvaný Kurzweil Reading Machine. Kurzweil Computer Products nakonec koupila firma Xerox, která se převodu do elektronické podoby chtěla dále věnovat.



Obrázek 3.1: Porovnání fontů OCR-A a OCR-B

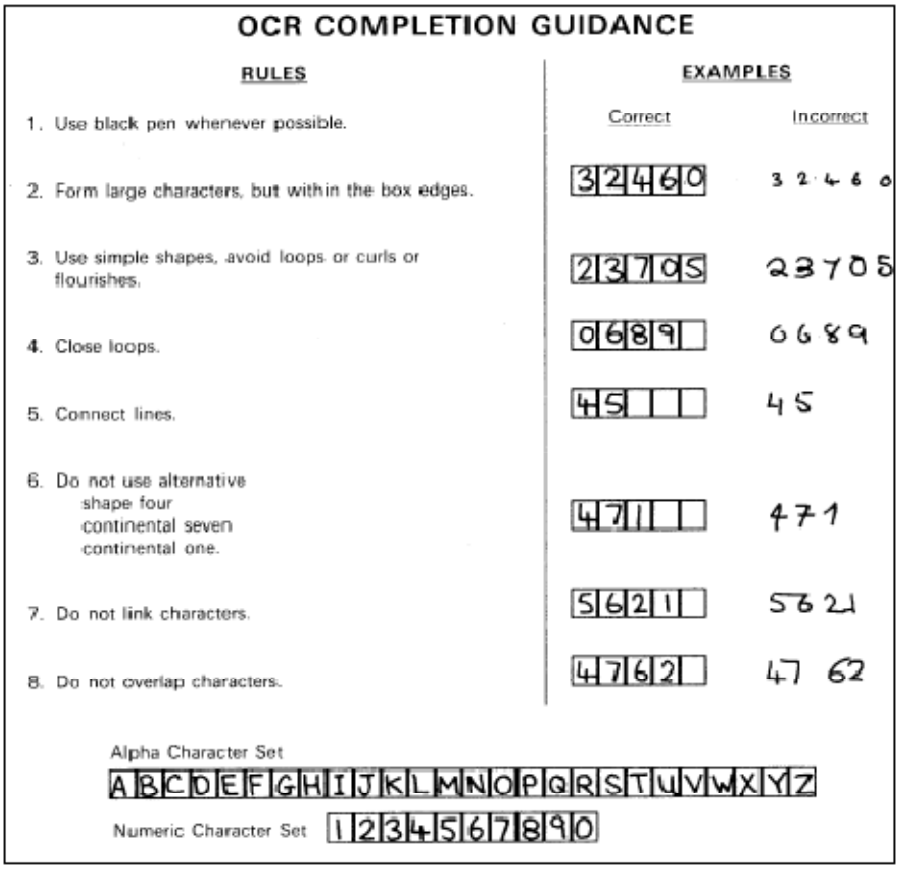
3.2 OCR dnes

Dnes je OCR software daleko dostupnější, než tomu bývalo kdysi. Existuje několik druhů softwaru jak na serverech, tak desktopové, ale zpravidla všechny vyhledávají text v obrázku a transformují ho s co možná nejméně chybami do upravovatelného formátu (TXT,DOC). Ty lepší softwary zvládnou převést i tabulky nebo zachovat formát písma. Dnes nejrozšířenější systém rozpoznávání tištěných znaků a kódů je čtečka čárových kódů. [11] Velké oblibě se dostává také QR kódům (Quick Response Code), které dokáží nést více informací než běžný čárový kód. Ukázku QR kódu lze vidět na obrázku 3.2.

Obrázek 3.2: QR kód

3.3 ICR - Intelligent Character Recognition

Dnes se můžeme setkat i s tzv. „inteligentními" OCR neboli ICR, které oproti klasickému OCR ovládají schopnost učit se a postupně se zlepšovat. Jsou tedy schopny uložit do své znalostní databáze předložený znak a následně ho standardně používat při rozpoznávání. Tento systém se používá především pro ručně psaný text, kdy se pomocí speciálních formulářů vkládají do databáze ručně psané znaky. Příklad takového formuláře můžete vidět na obrázku 3.3. Uživatel však většinou musí při rozpoznávání zasáhnout a označit, jestli byl daný znak rozpoznán kvalitně. V případě špatného převodu je zapotřebí, aby uživatel označil, kde se vyskytla chyba. Často se také musí přizpůsobit možnostem tohoto systému. Z důvodu rozdílnosti lidského rukopisu u každého jednotlivce však ICR nemůže fungovat globálně. [11][12]



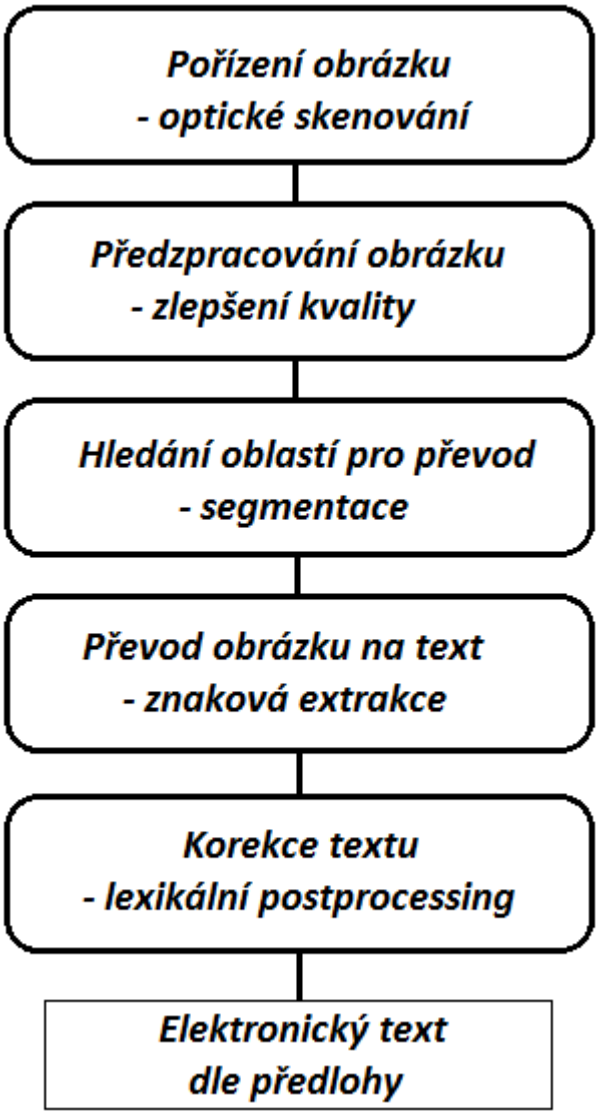
Obrázek 3.3: ICR formulář pro naučení ručně psaných znaků [12]

3.4 OMR - Optical Mark Recognition

Tato metoda je využívaná u speciálních formulářů (většinou dotazníky a testy), kde se vyhodnocuje prosvícení papírového podkladu. Při začernění určitého místa je průchod světla menší a následně jsou tato místa detekována a vyhodnocena. [12]

3.5 Fáze převodu

Krom samotného převodu tištěného textu do elektronické podoby se provádí ještě několik důležitých procesů, které ovlivňují samotnou kvalitu převodu. Jednotlivé kroky, tak jak jdou za sebou, jsou uvedeny na obrázku 3.4.



Obrázek 3.4: Fáze OCR

3.5.1 Pořízení obrázku

První fáze je pořízení zdrojového obrázku. Je to nejdůležitější část, od které se odvíjí kvalita převodu. Obrázek lze získat přímo z připojeného zařízení, např. skeneru, fotoaparátu nebo i nepřímo, kupříkladu stažením z internetu.

3.5.2 Předzpracování obrázku

Pořízený obrázek nemusí být vždy ideální pro převod. Může být různě barevný, otočený, obsahovat šum, nerovný text, mít špatnou kvalitu apod. Všechny tyto faktory zhoršují samotný

převod. Proto je nutné tyto obrázky upravit různými filtry a algoritmy (viz kapitola 3.6). Výsledný obrázek je pak vstupem pro OCR. [11]

3.5.3 **Převod**

Při samotném převodu se vstupní obrázek rozděluje na části podle souvislé oblasti pixelů, což se nazývá segmentace. Segmentace funguje na principu odlišnosti barev pozadí a znaků. Následuje tzv. „znaková extrakce" rozdělující rozpoznané částí na znaky, slova, řádky a bloky textu. Z těchto rozčleněných textů se nadále zjišťuje použitý typ písma (tj. font, formátování apod.), což usnadňuje práci následujícímu procesu nazvaným „klasifikace znaků". Tento proces vytváří pro jednotlivý znak vektor charakterizující daný znak. Porovnáním tohoto vektoru s databází se určí, o jaký znak se jedná.

3.5.4 **Korekce textu**

U většiny systémů následuje ještě korekce textu, tzv. „lexikální postprocessing", který porovnává skupiny znaků se svým jazykovým slovníkem a přiřadí této skupině slovo podle nejlepší shody. Výstupem převodu je text v elektronické podobě. [12]

3.6 Metody pro zlepšení vstupního obrazu

Následuje popis několika nejběžnějších metod, které se aplikují na vstupní obrázek před samotným zahájením převodu. Podrobnější popis těchto metod naleznete v odkazu [11].

3.6.1 **Převod obrázku na odstíny šedé**

Pro zjednodušení práce některým dalším metodám se převádí obrázek na odstíny šedé. Například některé metody pro hranové detekce nemusí zkoumat jednotlivé barevné složky zvlášť, ale pracují pouze s hodnotou jasu. Pro převod barevných složek modelu RGB na odstíny šedé se používají váhové koeficienty z důvodu rozdílné citlivosti lidského oka na tyto jednotlivé barevné složky. Vzorec pro převod je uveden následovně, kde R, G, B jsou hodnoty jednotlivých barevných složek a f je výsledná hodnota v odstínu šedé.

$$f = 0{,}299 * R + 0{,}587 * G + 0{,}114 * B \qquad (3.1)$$

Vzorec 3.1: Převod barevných složek na odstíny šedi

3.6.2 **Geometrické transformace**

Tato metoda se používá pro různé transformace či deformace obrázku. Základními a také nejpoužívanějšími transformacemi jsou afinní transformace, které zahrnují posunutí, otočení, změnu měřítka a zkosení. Nejčastěji se tyto transformace provádějí v homogenním souřadnicovém systému, kdy se používá jejich maticový zápis. Výhoda homogenního souřadnicového systému spočívá v možnosti vynásobit jednotlivé matice transformací a vytvořit tak jedinou matici, která se následně použije u všech bodů obrázku stejně. Příklad otočení souřadnic bodu okolo počátku souřadnicového systému můžete vidět ve vzorci 3.2, kde x a y jsou původní souřadnice bodu a x' a y' jsou nové souřadnice bodu otočené o úhel α.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.2}$$

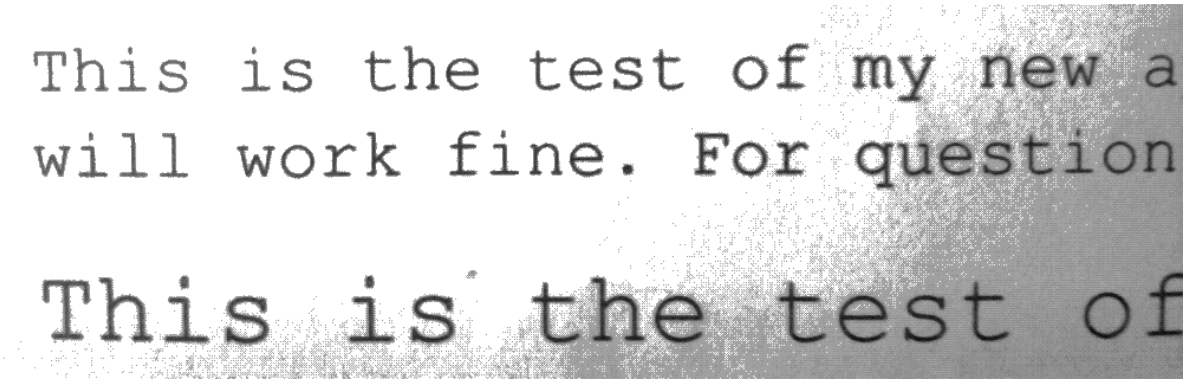
Vzorec 3.2: Transformace otočení souřadnic bodů okolo počátku souřadnicového systému

3.6.3 Jasové transformace

Tyto metody se používají pro změnu jasu všech obrazových bodů obrázku. Patří sem inverze barev, gama korekce, logaritmické transformace a roztažení kontrastu. U inverze barev se invertuje hodnota jasu každého bodu obrázku na stupnici od 0 do 255. Počítáme-li s bodem s hodnotou jasu 235, převede se na hodnotu jasu 20 a naopak. Gama korekce se používá pro úpravu jasu obrázku, který může obsahovat příliš tmavé nebo příliš světlé části. Logaritmické transformace slouží pro úpravu velkého rozsahu hodnot vzniklého při některých obrazových operacích. Tento rozsah by nebylo možné rozumně zobrazit, proto je nutné jej zkorigovat do menšího poměru mezi minimální a maximální hodnotou. Poslední metoda roztažení kontrastu se využívá při obrázcích, kde je minimální rozdíl mezi úrovněmi jasu jednotlivých prvků v obrázku, což je znesnadňuje odlišit a následně rozpoznat. Cílem je tedy mezi těmito prvky zvětšit kontrast.

3.6.4 Vyhlazování obrazu

Vstupní obrázky mohou obsahovat nežádoucí jevy, jako je například šum (ukázku šumu lze vidět na obrázku 3.5). Tyto nežádoucí jevy je třeba potlačit a na každý zvolit vhodnou metodu vyhlazování. Mezi tyto metody patří průměrování, mediánová filtrace a Gaussovo vyhlazování. Při průměrování se výsledná hodnota bodu bere jako průměr původní hodnoty bodu a hodnot nejbližších sousedů tohoto bodu. U mediánové filtrace se opět zkoumají hodnoty určitého bodu a jeho okolních bodů, z nichž se však nyní vybírá medián, který se následně určí jako nová hodnota tohoto bodu. Gaussovo vyhlazování je rozšíření metody průměrování, kdy se zavádí váha jednotlivých okolních bodů. Čím je okolní bod dále od zvoleného bodu, tím má jeho hodnota menší váhu a méně ovlivňuje výsledný bod. Počet a vzdálenost okolních bodů jsou omezeny. Metody průměrování a Gaussova filtrace však rozmazávají obrázek, což může být problém pro metodu detekující hrany.



Obrázek 3.5: Ukázka šumu v textovém obrázku

3.6.5 Detekce hran

Proces detekující hrany spočívá v hledání míst v obrázku, kde se vyskytují výrazné změny jasu. Pro detekci těchto změn se používají první a druhé derivace intenzity jasu. Tato metoda tedy

dokáže oddělit různě barevné prvky od sebe, což se využívá například při rozpoznávání textu na barevném pozadí. Mezi detektory hran patří Robertsův operátor, operátor Prewittové, Sobelův operátor, Cannyho hranový detektor, Laplaceův operátor a další.

3.6.6 Prahování

Při této metodě se obrázek převádí pouze do několika málo úrovní jasu, nejčastěji do dvou, a tím zobrazuje pouze určité části obrázku podle hodnoty jasu. Je určena mezní neboli prahová hodnota, která určuje, na jakou hodnotu se určitý bod obrázku převede. Vše co je pod prahovou hodnotou se nastaví na 0, vše ostatní na 1, popřípadě 255. Hodnotu prahu není vždy lehké určit, ovšem pro automatické nalezení existují již metody. Lze ji také posoudit například z histogramu, jenž znázorňuje zastoupení jednotlivých jasů v obrázku. Různé oblasti obrázku mohou mít různé hodnoty prahu, takové metodě se pak říká adaptivní prahování.

3.7 Metody OCR

Poté co vstupní obrázek projde různými algoritmy pro zlepšení, nastává samotné rozpoznávání znaků. Za tímto rozpoznáváním se skrývá opět několik algoritmů, jako jsou například algoritmy pro určení shody se šablonou (viz kapitola 3.7.1), pro popsání obrazce a jeho geometrických vlastností, anebo také algoritmy pro popis obrazců založených na invariantách (viz kapitola 3.7.2). Podrobnější popis těchto metod lze nalézt v odkazu [11].

3.7.1 Střední kvadratická chyba MSE (Mean Squared Error)

Tato metoda je určena pro nalezení nejlepší shody mezi vzorovými daty a jednotlivými rozpoznanými znaky. Vzorec pro výpočet je:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2, \tag{3.3}$$

kde x_i, $i = 1...n$, je řada n pixelů ze vzorového znaku a y_i, $i = 1...n$, je řada n pixelů z rozpoznávaného znaku. Čím je MSE nižší, tím je shoda jednotlivých pixelů vyšší. Tedy při nulové MSE je shoda absolutní. [13]

3.7.2 Obrazové invarianty

Invarianty znázorňují neměnící se vlastnosti stejných obrazců při jejich různé transformaci, jako je změna velikosti, otočení, zkosení apod. Rozpoznání touto metodou lze rozdělit na dva způsoby.

První používá k rozpoznání pouze hranice invariant, které podle obrysu objektu umí vytvořit kód řetězce nebo Fourierovy deskriptory, obojí popisující tvar objektu pomocí vektoru nebo čísla.

Druhý způsob používá celou oblast invarianty, kdy se získává číselný popis tvaru pomocí všech obrazových bodů objektu. Oproti metodě založené na hranicích objektu lze zde zjistit více informací z obrazce. Zde nejvíce používané invarianty jsou invarianty založené na momentech, kdy se opět vytváří číselná reprezentace objektu. [11]

3.8 Vliv typu písma na převod

Jak už bylo zmíněno, nejlepším možným fontem pro tuto technologii je standart OCR-A (viz obrázek 3.1). Avšak tento font nepodporuje některé české znaky, jako jsou např. "č, ř, ď, ť, ň". Co se týče ostatních fontů, jejich různorodost velice znesnadňuje rozpoznávání. Některé lze rozpoznat lépe, některé hůře.

To však není jediný problém při rozpoznávání. Mezi další patří špatně nasnímáný text jak už v nízké kvalitě, tak rotován o nějaký úhel. Dále také existují dvojce písmen připomínající jiné písmeno. Takový nejběžnější případ je dvojce "r n", které dohromady vypadají jako písmeno "m".

Na druhou stranu, když už máme kvalitně pořízený zdrojový text, dosahuje přesnost rozpoznání u latinky až 99% úspěšnosti a v současné době je to považováno za aplikačně vyřešitelný problém. 100% úspěšnost je možná pouze při zásahu člověka. Co se týče složitějších znaků, jako jsou ručně psané texty, znaková písma (např. čínština) apod., není tato problematika stále vyřešena. Pro ručně psané znaky takové softwary sice existují, ale dosahují přesnosti rozpoznání kolem 85%. [11]

4 Knihovny pro OCR v programovacím jazyce Java

OCR knihovny se dělí na dva základní druhy. Jedny poskytují třídy pro práci s internetovými servery (tzv. cloudy) a druhé provádí převody přímo na klasických desktopových a mobilních zařízeních (dále jako převádějící knihovny). Jádro většiny z nich však není napsáno v programovacím jazyce Java. V Javě poskytují pouze obalovací třídy pro práci s tímto jádrem.

4.1 Převody v cloudu

Výhoda těchto knihoven spočívá v různorodosti platforem a programovacích jazyků. Tyto knihovny obsahují pouze třídy pro komunikaci a práci s cloudem, neprovádí samotný převod. Posílají data na server, který provádí hardwarově náročné operace převodu a následně pošle zpět výsledek převodu. Je ideální pro méně výkonná zařízení, na kterých by trval převod dlouho. Takhle se zařízení stará jenom o transfer vstupních a výstupních dat převodu. Nevýhodou však je nutné připojení k internetu.

4.2 Převody na zařízeních

Logicky jsou převádějící knihovny opakem předchozích. Tyto knihovny si vývojář může přidat do svých softwarů a využívat je kdykoliv bez nutnosti připojení k internetu. Nevýhodou však může být pomalejší převod u méně výkonnějších zařízení.

4.3 Knihovny

Následuje výpis nalezených OCR knihoven pro programovací jazyk Java. Prvních osm knihoven je placených a jsou řazeny sestupně od nejlepší po nejhorší. Dále jsou zde dvě neplacené knihovny.

4.3.1 Abby FineReader Engine

Dnes asi nejlepší převádějící knihovna pro OCR, ICR, OMR, čárové kódy, formuláře a další. Nabízí největší podporu jazyků a to 198 jazyků pro OCR, 113 jazyků pro ICR a navíc také možnost si vytvářet další uživatelské jazyky. Podporuje všechny dnešní běžné formáty obrázků pro vstup a textů pro výstup. Navíc umožňuje použít i formát PDF pro vstup i výstup. Dále je tato knihovna schopna rekonstrukce původní logické struktury a formátování dokumentu či tabulkové struktury, rozpoznávat vertikální text, magazínové stránky a další. Jedná se o placenou knihovnu, kterou si však lze, pokud jste firma, na vyžádání vyzkoušet v časově omezené verzi. Získala i několik ocenění za konverzi. Kompletní přehled o knihovně lze zjistit na stránce http://www.abbyy.cz/products/sdk/.

4.3.2 Abby Mobile OCR Engine 4.0

Je upravená předchozí knihovna pro méně výkonné přenosné zařízení. Jelikož pořízení snímku z fotoaparátu smartphonu je horší než skenování, nabízí knihovna třídy pro různé korekce těchto snímků. Podporuje 21 základních jazyků včetně češtiny a dále 40 přídavných jazyků. Má malé paměťové nároky, pouze 8 MB pro ROM a 10 MB pro RAM. Tato knihovna však není čistě v Javě, a proto se musí použít obalovací třídy a Android NDK. Více lze zjistit na stránce http://www.abbyy.cz/products/sdk/.

4.3.3 Abby Cloud OCR SDK

Vysoce přesná on-line knihovna založena opět na stejné technologii jako FineReader Engine. Od FineReader Engine se liší pouze používáním v cloudu. Poskytuje webové API, přes které lze pracovat s touto knihovnou. Je zdarma pro vývoj a testování, platí se pouze za komerční používání. Více informací lze zjistit na stránce http://ocrsdk.com.

4.3.4 Leadtools OCR SDK

Firma Leadtools nabízí jedny z lépe vybavených knihoven poskytující převod jak na zařízeních, tak i v cloudu. Podporuje přes 30 jazyků, 150 obrázkových formátů pro vstup a mnoho textových formátů pro výstup (včetně PDF, DOC, DOCX, XML a dalších). Umožňuje automatické opravy vstupních obrázků a kontrolu pravopisu výstupního textu. Dále rozpoznává formátování dokumentu a tabulek, grafiku, styl písma, čárové kódy a další. Jedná se o placenou knihovnu, kterou si lze vyzkoušet po dobu 60 dní. Ukázka použití pro Android je uvedena v příkladu 4.1. Více informací na stránce http://www.leadtools.com/sdk/ocr/default.htm.

```
Bitmap bitmap =
BitmapFactory.decodeResource(getResources(),R.drawable.test);
RasterImage rasterImage =
RasterImageConverter.convertFromBitmap(bitmap,
ConvertFromImageOptions.NONE);
OcrEngine ocrEngine =
OcrEngineManager.createEngine(OcrEngineType.Advantage);
ocrEngine.startup(null, "", null,
this.getApplicationInfo().dataDir);
OcrDocument document =
ocrEngine.getDocumentManager().createDocument();
OcrPage ocrPage = document.getPages().addPage(image, null);
String text = ocrPage.recognizeText(this);
```

Příklad 4.1: Ukázka použití knihovny Leadtools OCR SDK

4.3.5 Digital Syphon Sonic Imagën

Další z placených knihoven, která však oproti ostatním nabízí navíc rozpoznání vektorových obrázků a obrázků založených na matematických výrazech. Dále zvládne automatickou detekci rozložení dokumentu, vyhledávání loga, detekci obličeje z obrázku a detekci miniatur z videa. Pro loga a obličeje se musí uvést vzorová data. Podporuje běžné obrázkové formáty pro vstup (PNG, JPG, GIF, TIFF, PDF), dva textové formáty pro výstup (TXT, XML) a všechny základní jazyky. Více informací lze zjistit na stránce http://www.digitalsyphon.com/technologies_sonicimagen.asp?contentpage=technologies_sonicimajen&bodyid=technologies&technologies=technologies.

4.3.6 Asprise OCR SDK v4.0

Další převádějící knihovna nabízející dobrý a rychlý převod běžných obrázkových formátů, rozpoznávání formátování dokumentu a jednoduché použití. Taktéž rozpoznává různé druhy čárových kódů, obrázky typu TIFF a PDF dokumenty. Jedná se o placenou knihovnu, kterou si však lze zdarma

vyzkoušet. Ukázka použití je uvedena na příkladu 4.2. Více informací o této knihovně lze nalézt na stránce http://asprise.com/product/ocr/index.php?lang=java.

```
File file = new File("D:/testimage.png");
System.load("C:/Windows/ILU.dll");
System.load("C:/Windows/AspriseOCR.dll");
System.load("C:/Windows/DevIL.dll");
BufferedImage image=null;
     try {
          image = ImageIO.read(file);
          } catch (IOException e) {
               e.printStackTrace();  }
String s = new OCR().recognizeCharacters(image);
System.out.println("\n---- RESULTS: ------- \n" + s);
```

Příklad 4.2: Ukázka použití knihovny Asprise OCR SDK v4.0

4.3.7 Saaspose.OCR SDK for Java

Je knihovnou nabízející převod v cloudu. Podporuje pouze dva obrázkové formáty pro vstup (BMP, TIFF), dva textové formáty pro výstup (XML, JSON), čtyři jazyky (Angličtina, Ruština, Francouzština a Španělština) a šest základních fontů (Arial, Times New Roman, Courier New, Verdana, Tahoma a Calibri). Dále umí rozpoznat styl písma, skenovat celý dokument nebo jenom jeho vybrané oblasti a skenovat otočené dokumenty. Pro plné využití je třeba klíč od výrobce, který si je třeba dokoupit. Lze si však tuto knihovnu vyzkoušet i zdarma. Ukázka použití lze vidět na příkladu 4.3. Více informací lze zjistit na stránce http://saaspose.com/api/ocr.

```
//specifikace URI produktu
com.saaspose.common.Product.setBaseProductUri("http://api.saaspose.c
om/v1.0");
//specifikace aplikačního klíče a SID
com.saaspose.common.SaasposeApp.setAppKey("**********");
com.saaspose.common.SaasposeApp.setAppSID("**********");
//Inicializace extractoru
com.saaspose.ocr.Extractor extractor = new
com.saaspose.ocr.Extractor();
//Získání textu a PartsInfo
OCRResponse response = extractor.ExtractText("c:\\temp\\test.bmp",
LanguageName.English, true);
```

Příklad 4.3: Ukázka použití knihovny Saaspose.OCR SDK

4.3.8 Aspose.OCR for Java

Knihovna určená pro převod na zařízeních i ve webových aplikacích. Umí zjistit text z obrázku a rozpoznat styl písma, ale podporuje pouze tři typy fontů (Arial, Tahoma a Times New Roman) a dva typy obrázků (BMP, TIFF). Ukázka použití je uvedena v příkladu 4.4. Více informací lze zjistit na stránce http://www.aspose.com/java/ocr-component.aspx.

```
OcrEngine ocr = new OcrEngine(ResourcesSource.BINARY_ZIP_FILE,
"./resources/resources.zip","xmlFilename","arialAndTimesAndCourierRe
gular.xml.bin";
ocr.getConfig().setNeedRotationCorrection(false);
ocr.setImage(File("./sample.bmp"));
ILanguage language = Language.load("english");
ocr.getLanguages().addLanguage(language);
try {      if (ocr.process()) {
           System.out.println("\ranswer -> " + ocr.getText());   }
} catch (Exception e) {
     e.printStackTrace();  }
```

Příklad 4.4: Ukázka použití knihovny Aspose.OCR SDK

4.3.9 **Tesseract - ocr**

Velice přesná open source knihovna od Googlu srovnatelná s mnoha komerčními produkty. Jádro je napsané v programovacích jazycích C/C++, existuje však několik obalovacích tříd v programovacím jazyce Java, které tuto knihovnu využívají, jako jsou např. Tess4J (ukázka použití je uvedena v příkladu 4.5), Tess-two, tesseract-android-tools a další. Tesseract v základu podporuje pouze obrázky TIFF formátu, ale používá se v kombinaci s knihovnou Leptonica, která přidává další obrázkové formáty. Dále umí rozpoznat rozložení stránky a podporuje přes 60 jazyků a PDF formát. Existuje také mnoho knihoven poskytující grafické rozhraní k této knihovně. Více informací lze zjistit na stránce https://code.google.com/p/tesseract-ocr/.

```
public class TesseractExample {
    public static void main(String[] args) {
        File imageFile = new File("eurotext.tif");
        Tesseract instance = Tesseract.getInstance();
        try {
            String result = instance.doOCR(imageFile);
            System.out.println(result);
        } catch (TesseractException e) {
            System.err.println(e.getMessage());
        } } }
```

Příklad 4.5: Ukázka použití knihovny Tesseract

4.3.10 **Java OCR**

Tato převádějící knihovna slouží čistě pro porovnání znaků z obrázku s již předehranými vzorovými znaky. Vzorové znaky však knihovna neobsahuje a vývojář si je musí vytvořit sám. Knihovna je celá napsaná v programovacím jazyce Java a obsahuje pouze třídy pro filtrování vstupního dokumentu, převod do stupňů šedi, rozčlenění dokumentu na řádky a následně na znaky, porovnání znaků s již předehranými znaky a vytvoření nejlepší shody s textem. Tuto knihovnu vytvořil Ron Cemer a poskytnul ji pod BSD licencí, tedy jako volně šiřitelnou. Díky tomu doznala

několik modifikací od dalších vývojářů, kteří si ji upravili pro své potřeby. Jeden z nich je i Dmitry Korotkov, který do knihovny přidal třídy pro mobilní aplikace a nazval ji Receipt-ocr.

Já jsem danou knihovnu upravil taktéž pro tuto bakalářskou práci. Ukázku použití můžete vidět na příkladu 4.6. Více informací o knihovně Java OCR lze nalézt na http://roncemer.com/software-development/java-ocr/.

```
private HashMap<Character, ArrayList<TrainingImage>> trainingImages;
trainingImages = new HashMap<Character, ArrayList<TrainingImage>>();
Bitmap b = BitmapFactory.decodeResource(getResources(),
R.drawable.arial);
TrainingImageLoader loader = new TrainingImageLoader();
loader.load(b, new CharacterRange('!', '~'), trainingImages);
OCRScanner ocr = new OCRScanner();
ocr.addTrainingImages(trainingImages);
Bitmap test= BitmapFactory.decodeResource(getResources(),
R.drawable.test);
String text = ocr.scan (test, 0, 0, 0, 0, null);
System.out.println(text);
```

Příklad 4.6: Ukázka použití knihovny Java OCR

5 Mobilní aplikace

Při samotném vývoji mobilní aplikace se setkáme s několika problémy. Jeden z nich je kompatibilita aplikace na různých zařízeních. Každé zařízení má jiný displej i hardwarové vybavení, takže na každém zařízení bude fungovat aplikace rozdílně. Vytváření klasických formulářových aplikací je jednodušší, než vytváření těch herních. U formulářových aplikací se používají již vytvořené prvky (tlačítka, layouty, textová pole apod.), které se automaticky přizpůsobují pro jednotlivé rozlišení displejů. Kdežto u herních aplikací si tyto prvky vytváří sám vývojář a může se stát, že se zobrazí ve špatném poměru, tedy roztáhlé nebo naopak zúžené. Další problém může nastat při operacích náročných na paměť, která méně výkonná mobilní zařízení nemusí zvládnout.

5.1 Softwarové vybavení

Pro vývoj byl použit softwarový nástroj Eclipse Juno s nainstalovanými Android Development Toolkit pluginem a Android SDK revize 21.1. Jako programovací jazyk byl použit jazyk Java uzpůsobený pro možnosti Androidu. Pro některé grafické tlačítka a ikonu aplikace byl použit modelovací nástroj Blender. Veškeré diagramy jsou vytvořeny v programu Microsoft Office Word 2007.

5.2 Hardwarové vybavení

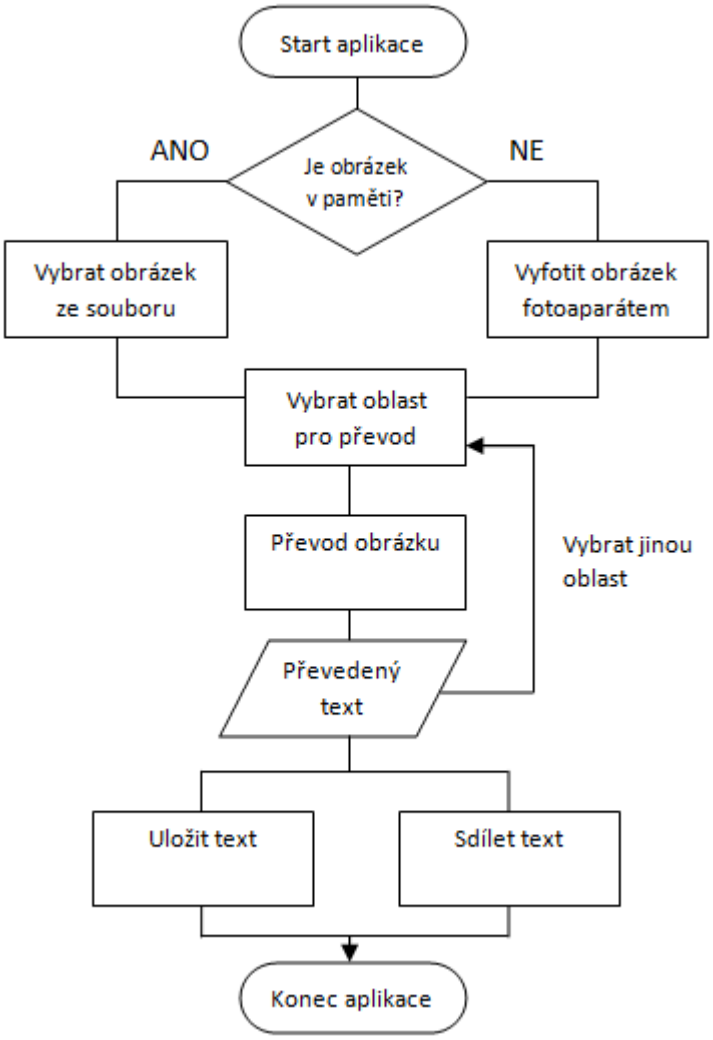
Aplikace byla testována na mobilním telefonu Sony Xperia P s verzí Androidu 4.0.4, s verzí jádra 3.0.8, pamětí RAM 1024MB a procesorem ST-Ericsson U8500 (2 x Cortex-A9 1000MHz). Dále byla vyzkoušena na emulátorech nastavených pro různé typy mobilních zařízení.

5.3 Požadavky pro běh

Pro rozběhnutí této aplikace je požadovaná minimální verze Androidu 2.3.3 nazývaná též Gingerbread. Dále je zapotřebí, aby mobilní zařízení obsahovalo fotoaparát podporující funkci automatického zaostřování neboli autofocus. Jako minimální rozlišení fotoaparátu je doporučeno 5 Mpx při focení dokumentů formátu A4, což odpovídá hustotě vstupního obrazu kolem 230 DPI. Jako optimální hustota vstupního obrázku pro převod se udává 300 DPI, to odpovídá zhruba 8 Mpx při focení formátu A4. Pro úplné a nejlepší využití aplikace doporučuji nainstalovat aplikace Total Commander a Překladač Google z internetového obchodu Google Play.

5.4 Seznámení s aplikací

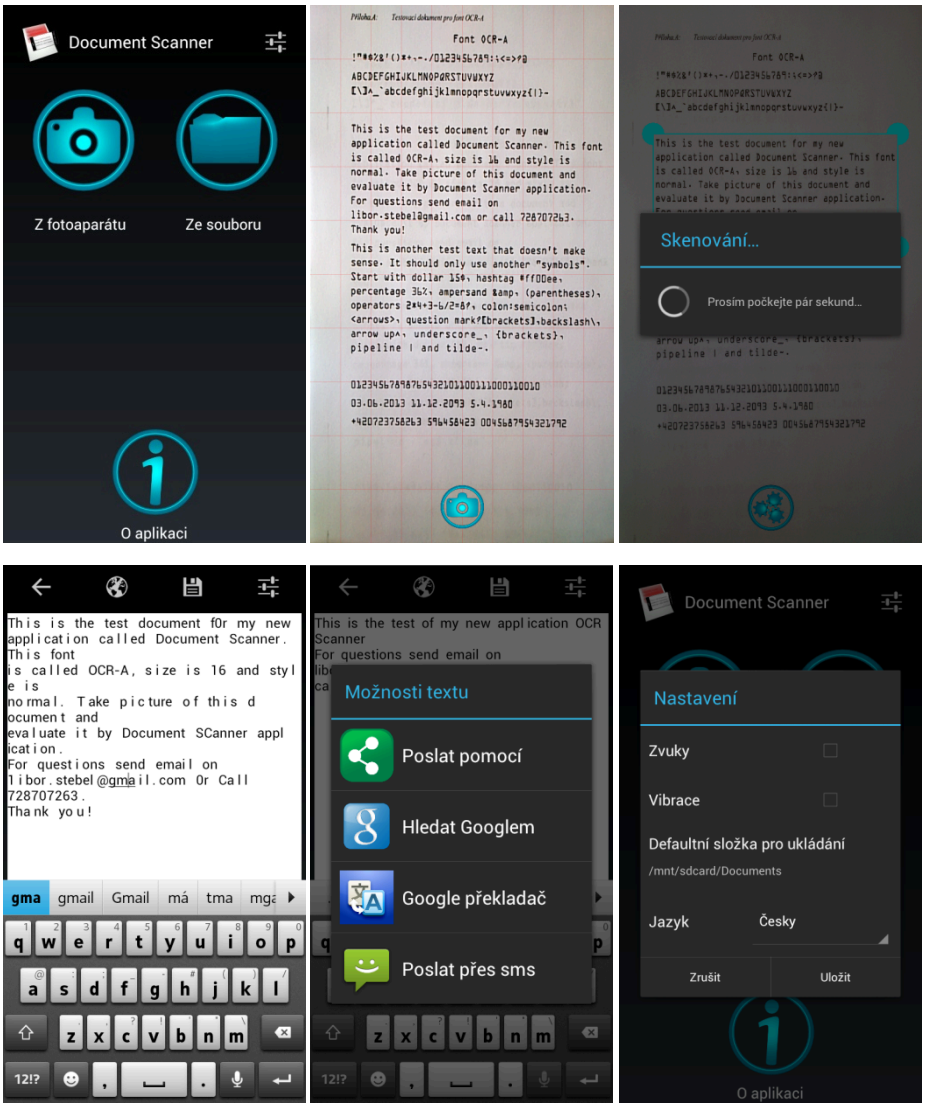
Návrh celé aplikace je založen na jednoduchosti a rychlosti použití. Zároveň odpovídá po grafické i logické stránce dnešním aplikacím. Jsou zde použity ikony pro základní operace jako nastavení a uložení pořízené z oficiální stránky Androidu. Na obrázku 5.1 můžete vidět vývojový diagram aplikace a na obrázku 5.2 pak ukázky z aplikace.



Obrázek 5.1: Vývojový diagram

Po spuštění aplikace se uživateli zobrazí úvodní obrazovka. Zde má uživatel na výběr čtyři možnosti. První je tlačítko zobrazující při kliknutí informace o aplikaci. Druhá možnost je tlačítko nastavení, kde je uživatel schopen vybrat ze dvou jazyků (čeština, angličtina), vypnout nebo zapnout zvuková a vibrační upozornění a nakonec se zde dozví defaultní cestu pro ukládání souborů z této aplikace. Poslední dvě možnosti slouží k výběru zdrojového obrázku pro převod. Jedno tlačítko pro pořízení z fotoaparátu a druhé tlačítko pro pořízení ze souboru.

Při výběru pořízení z fotoaparátu se spustí obrazovka s možností vyfotit jakýkoliv obrázek. Pokud se uživatel dotkne obrazovky, fotoaparát zaostří. Pokud stiskne softwarové nebo hardwarové tlačítko pro vyfocení, fotoaparát opět zaostří a pořídí snímek. Následně se spustí další obrazovka pro práci s tímto snímkem. Při výběru pořízení ze souboru se zobrazí vhodné souborové manažery pro výběr obrázku ze souboru. Po vybrání obrázku se spustí další obrazovka pro práci s tímto obrázkem.



Obrázek 5.2: Ukázky z aplikace

Na další již zmíněné obrazovce má uživatel možnost vybrat oblast, která se má převést. Po načtení všech souborů potřebných pro převod se zobrazí tlačítko pro zahájení převodu. Při stisku tohoto tlačítka se zobrazí informační okénko znázorňující proces převádění.

Při dokončení převodu se zobrazí poslední obrazovka, kde má uživatel několik možností, jak pokračovat s převedeným textem. Může text libovolně upravit nebo přepsat. Dále je zde tlačítko zpět, tlačítko pro sdílení textu, tlačítko pro uložení textu a nakonec opět tlačítko nastavení popsané už dříve. Při stisku tlačítka zpět se uživatel vrátí na předchozí obrazovku, kde si může vybrat jinou oblast a znovu převést. Pokud uživatel stiskne ikonu zeměkoule, zobrazí se několik dalších možností, jak sdílet text. Je zde možnost poslat tento text pomocí emailu, sms a bluetooth. Dále je možné text hledat pomocí vyhledávače Google, nebo dokonce přeložit pomocí aplikace Překladač Google, která ovšem musí být nainstalovaná. Při stisku ikony znázorňující ukládání se zobrazí uživateli nabídka pro zadání názvu souboru. Pokud uživatel zadá název a ten následně potvrdí tlačítkem „Ok", soubor se uloží do složky uvedené v nastavení. Pokud ovšem má uživatel nainstalovanou aplikaci Total Commander, může si zvolit, kam soubor uložit.

5.5 Popis jednotlivých aktivit a tříd

Následuje podrobný popis jednotlivých aktivit a tříd, které byly vytvořeny pro tuto aplikaci. Veškeré zdrojové kódy této aplikace si lze prohlédnout na přiloženém CD.

5.5.1 Aktivita MainActivity

Za úvodní stránkou se skrývá aktivita MainActivity, která řeší čtyři základní úkoly. První dva jsou zobrazení dialogů pro nastavení a pro informace o aplikaci. Obojí je řešeno pomocí třídy AlertDialog.Builder, která nabízí metody pro určení nadpisu, obsahu a tlačítek. Pro jednoduchý dialog o aplikaci, kde je v obsahu jenom text, to stačí. Pro nastavení je však vytvořen vlastní obsah, který je pomocí třídy LayoutInflater převeden na instanci třídy View a následně nastaven jako obsah dialogu. Tento obsah se ukládá pomocí třídy SharedPreferences.Editor jako jednoduché datové prvky do paměti telefonu, tudíž se uchovávají i po ukončení aplikace.

Další dva úkoly souvisí s pořízením zdrojového obrázku pro převod. První z nich je pořízení z kamery mobilního zařízení a druhý pořízení z paměti telefonu. Při volbě pořízení z kamery je vytvořena instance třídy Intent, kde je v konstruktoru nastavená aktivita pro práci s kamerou nazvaná MyCamera (viz kapitola 5.5.2). Následně je tato instance spuštěna pomocí metody `startActivity`. Při volbě pořízení ze souboru je opět vytvořena instance třídy Intent, tentokrát je však v konstruktoru nastavená příslušná akce pro výběr souboru, tedy `Intent.ACTION_GET_CONTENT`. Dále je určen typ souboru pomocí metody `setType(image/*)`, zde tedy obrázkový. Následně je tato instance spuštěna pomocí metody `startActivityForResult`, která při výběru obrázku zavolá metodu `onActivityResult`. V této metodě se získá cesta k vybranému souboru a následně se z této cesty vytvoří instance třídy Bitmap pomocí `BitmapFactory.decodeFile(path)`. Z této instance je zjištěna orientace obrázku pomocí třídy ExifInterface a následně je převedena na pole bytů pomocí třídy ByteArrayOutputStream. Toto pole bytů je uloženo prostřednictvím třídy FileOutputStream do složky pro cache soubory aplikace získanou metodou `getCacheDir()`. Výsledná cesta k tomuto souboru a orientace obrázku jsou poslány pomocí nové instance třídy Intent další aktivitě. Tato instance má v konstruktoru nastavenou aktivitu pro úpravu obrázku EditImageSize a následně je spuštěna pomocí metody `startActivity`. Takto složitý proces převodu obrázku na pole bytů je z důvodu stejné formy vstupních dat do aktivity EditImageSize z obou aktivit MainActivity i MyCamera.

5.5.2 Aktivita MyCamera

V pozadí druhého způsobu pořízení obrázku, nyní kamerou, je aktivita nazvaná MyCamera. Pro možnost promítání snímků z kamery na displej je vytvořena třída CameraPreview. CameraPreview rozšiřuje třídu SurfaceView, tzn. můžeme ji nastavit jako pozadí aktivity. Dále tato třída drží instanci kamery předané v konstruktoru a udává, kdy má kamera promítat a kdy ne pomocí metod `surfaceCreated`, `surfaceDestroyed a surfaceChanged`. Samotná aktivita MyCamera pořízuje instanci hardwarové kamery a nastavuje jí paramtery pomocí třídy Camera.Parameters. Tuto instanci také předává již zmíněné třídě CameraPreview. Dále jsou zde dvě instance třídy AutoFocusCallback určeny pro zaostření kamery, kdy každá reaguje na jinou akci od uživatele. Při dotyku uživatele na displej se volá metoda `onDisplayTouch` spouštěcí první AutoFocusCallback. Druhý AutoFocusCallback je volán při stisku softwarového tlačítka kamery

v metodě `onCapture` nebo hardwarového tlačítka kamery v metodě `onKeyDown`. Zde však po zaostření následuje pořízení snímku a volání speciální metody `onPictureTaken` na instanci třídy PictureCallback. Tato metoda obsahuje v jednom z parametrů vyfocený obrázek ve formě pole bytů. Toto pole bytů je uloženo pomocí třídy FileOutputStream do složky pro cache soubory aplikace získanou metodou `getCacheDir()`. Výsledná cesta k tomuto souboru a orientace obrázku jsou poslány pomocí nové instance třídy Intent další aktivitě. Tato instance má v konstruktoru nastavenou aktivitu pro úpravu obrázku EditImageSize a následně je spuštěna pomocí metody `startActivity`.

5.5.3 Aktivita EditImageSize

Po pořízení obrázku jak z kamery, tak ze souboru následuje aktivita nazvaná EditImageSize. O její vzhled se stará třída nazvaná CropImageView, která rozšiřuje třídu View. V třídě CropImageView se nahrává pole bytů ze souboru představující pořízený obrázek a následně se dekóduje na instanci třídy Bitmap pomocí metody `BitmapFactory.decodeByteArray`. Tento obrázek je však zmenšen podle rozměrů displeje, aby zbytečně nezabíral paměť. Dále se vytváří instance třídy Canvas, která v metodě `onDraw` vykresluje tento obrázek jako první v pořadí. Následuje vykreslení obdelníku a posuvníků pro výběr oblasti. Dále je zde metoda `onTouchEvent` reagující na dotyky a posuvy uživatele na displeji. Při dotyku se zjišťuje, jestli souřadnice odpovídají jednomu z posuvníků. Pokud ano, nastavují se mu nové souřadnice podle pohybu na displeji. Pokud ne, neděje se nic. Tato třída pak obsahuje metody vracející souřadnice těchto posuvníků pro určení oblasti, kterou uživatel zvolil.

Při startu aktivity EditImageSize se spustí načítání training dat v novém vlákně pomocí třídy MyAsyn rozšiřující třídu AsyncTask. Je tedy možné při načítání již manipulovat s posuvníky. Pro samotné vytváření training dat je použita třída TrainingImageLoader. Po skončení načítání se zobrazí na displeji tlačítko pro zahájení převodu. Pokud uživatel stiskne toto tlačítko, vytvoří se nové vlákno, ve kterém poběží převod. V tomto vlákně se vytvoří instance třídy OCRScanner patřící do Java OCR knihovny a následně se jí přiřadí training data pomocí metody `addTrainingImages`. Dále se načte pole bytů ze souboru představující pořízený obrázek pomocí třídy FileInputStream a toto pole bytů se převede na instanci třídy Bitmap pomocí metody `BitmapFactory.decodeByteArray`. Nakonec se zjistí souřadnice posuvníků ze třídy CropImageView a společně s instancí třídy Bitmap se předají jako parametry metodě `scan` vyvolanou instancí třídy OCRScanner. Tato metoda vrací textový řetězec, který je poslán pomocí nové instance třídy Intent další aktivitě. Tato instance má v konstruktoru nastavenou aktivitu pro práci s textem EditScannedText a následně je spuštěna pomocí metody `startActivity`.

5.5.4 Aktivita EditScannedText

Za obrazovkou pracující s rozpoznaným textem běží poslední aktivita nazvaná EditScannedText. Pokud jsou v nastavení povoleny zvuky a vibrace, spustí se obojí při startu této aktivity za účelem informovat uživatele o dokončení převodu. Zvuk je spouštěn pomocí tříd RingtoneManager a Ringtone a vibrace jsou spouštěny pomocí třídy Vibrator. Text získaný z předchozí aktivity se nastaví instanci třídy EditText, která představuje textové pole na obrazovce. V tomto textovém poli si uživatel může text upravit podle vlastních potřeb. Dále jsou zde čtyři tlačítka. Pokud uživatel stiskne šipku zpět, zavolá se metoda `finish()` ukončující tuto aktivitu

a aplikace přejde zpět k přechozí aktivitě výběru oblasti pro převod. Pokud uživatel stiskne ikonu zeměkoule, zobrazí se dialog pro další možnosti textu vytvořený pomocí třídy AlertDialog.Builder. Tento dialog má jako obsah nastavenou instanci třídy ListView upravenou pomocí vytvořené třídy MyAdapter, která rozšiřuje třídu ArrayAdapter. Pro reakci na tento obsah implementuje tato aktivita onItemClickListener a tedy obsahuje metodu `onItemClick`, která se volá při stisku jakékoliv z možností v dialogu. Tyto možnosti nabízí sdílení textu pomocí emailu, bluetooth, SMS, dále hledání textu pomocí Google vyhledavače a přeložení do jiného jazyka pomocí aplikace Překladač Google. Všechny aplikace se spouští vytvořením a následným spuštěním instance třídy Intent avšak pokaždé s jiným typem akce a nastavením. Tyto akce jsou `ACTION_SEND` pro email a bluetooth, `ACTION_WEB_SEARCH` pro hledání pomocí Google vyhledavače a `ACTION_VIEW` pro SMS a přeložení pomocí Google překladače. Dalším tlačítkem je ikona diskety znázorňující uložení textu do souboru. Po stisknutí tohoto tlačítka se vytváří instance třídy Intent s nastavenou akcí `ACTION_PICK` sloužící pro výběr aplikací, které umí vybrat složku či soubor ze stromové struktury telefonu. Tato instance třídy Intent je následně spuštěna pomocí metody `startActivityForResult` a pokud je příslušná aplikace nainstalovaná v zařízení, spustí se. Po výběru cesty k uložení se volá metoda `onActivityResult`, kde se kontroluje správnost zadání a nakonec je text uložen do souboru. Pokud však takto vhodná aplikace není nainstalována v zařízení, je vyhozena vyjímka `ActivityNotFoundException`, ve které se spustí dialog pro zadání názvu souboru. Po zadání názvu a potvrzení tlačítkem „Ok" je soubor uložen do defaultní cesty uvedené v nastavení. Poslední tlačítko vyvolává dialog s nastavením, který byl již popsán v kapitole MainActivity.

5.6 Použitá OCR knihovna

V aplikaci byla použita OCR knihovna Java OCR. Z dvou volně dostupných knihoven byla vybrána z toho důvodu, že je napsaná celá v programovacím jazyce Java na rozdíl od knihovny Tesseract. Ostatní knihovny nepřicházely v úvahu jednak kvůli své ceně a jednak kvůli omezené době při používání zkušebních verzí. Jelikož knihovna Java OCR neobsahuje žádné training data, musela být vytvořena vlastní, která jsou popsána v kapitole 5.6.3. Další komplikace vznikly s jinou kompatibilitou programovacího jazyka Java pro OS Android než pro klasické desktopové OS. Knihovna totiž obsahovala prvky, které Java pro Android neobsahuje. V knihovně tedy musely být přepsány některé třídy a metody. Bylo přidáno také adaptivní prahování (viz kapitola 3.6.6) pro lepší kvalitu vstupního obrázku.

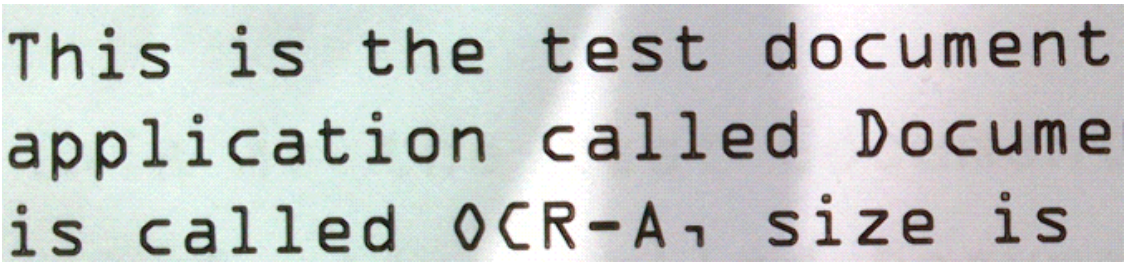
5.6.1 Předzpracování obrázku

Před samotným převodem, je vstupní obrázek rasterizován na jednotlivé pixely a upravován několika metodami pro zlepšení kvality obrázku. Nejprve je obrázek převeden na odstíny šedi (viz kapitola 3.6.1), poté je použito adaptivní prahování (viz kapitola 3.6.6) a nakonec je použito Gaussovo vyhlazování pro filtraci obrázku (viz kapitola 3.6.4). Na obrázku 5.3 můžete vidět obrázek před aplikací těchto metod a na obrázku 5.4 po aplikaci.

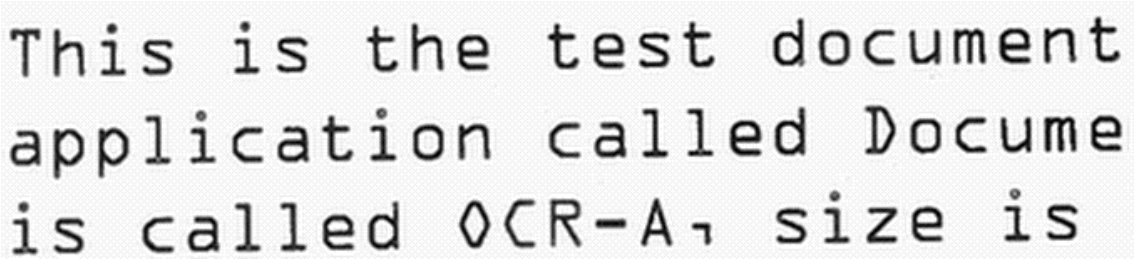
5.6.2 Převod

Po předzpracování vstupního obrázku tato knihovna nejprve roztřídí rasterizovaný obrázek na řádky a následně tyto řádky roztřídí na jednotlivé znaky. Roztřízené znaky si uchovává jako pole

pixelů a následně je porovnává s training daty, které jsou rovněž uloženy ve formě pixelů. Nakonec knihovna určuje nejlepší shodu rozpoznávaného znaku s training daty pomocí metody, která se nazývá Střední kvadratická chyba, již popsaná v kapitole 3.7.1.



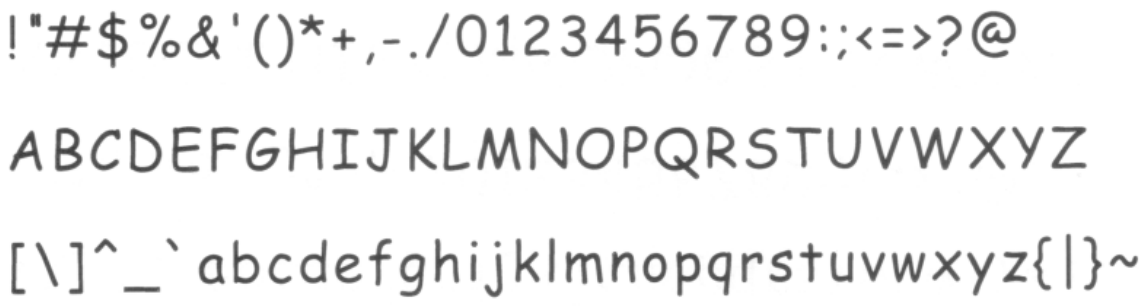
Obrázek 5.3: Původní vstupní obrázek před zpracováním



Obrázek 5.4: Zpracovaný vstupní obrázek

5.6.3 Training data

Jak už bylo zmíněno, pro aplikaci bylo vytvořeno několik training dat pro různé druhy fontů. Obsahují tyto fonty: OCR-A, OCR-B, Trebouchet MS, Lucida Console, Verdana, Comic Sans MS, Arial a Times New Roman. Pro poslední tři zmíněné fonty jsou nahrány i znaky s háčky a čárkami. Ukázku training dat pro font Comic Sans MS můžete vidět na obrázku 5.5.



Obrázek 5.5: Ukázka training dat pro font Comic Sans MS

5.6.4 Formáty vstupních obrázků

Knihovna Java OCR podporuje několik vstupních obrázkových formátu. Tyto formáty jsou odvozeny od třídy Bitmap, která se v této knihovně stará o práci s obrázky. Třída Bitmap a tedy i celá aplikace podporuje formáty obrázku JPEG, GIF, PNG a BMP. Od verze Androidu 4.0 a vyšších

podporuje třída Bitmap také formát WEBP, ale jelikož je tato aplikace dostupná od verze 2.3.3, není tento formát v aplikaci podporován.

5.7 Testování

Vytvořená aplikace byla otestována pro všechny nahrané typy fontů. Pro každý font byl vytvořen testovací dokument ve formátu A4 se vzorovým textem. Tento text simuluje použití všech nahraných znaků pro daný font. Tyto dokumenty si lze prohlédnout v přílohách.

Vstupní obrázky testovacích dokumentů byly pořizovány dvěma způsoby a navzájem se porovnávaly úspěšnosti rozpoznání znaků. První způsob pořízení spočíval ve vyfocení dokumentů pomocí vytvořené aplikace, tedy integrovaným fotoaparátem mobilního zařízení. Podruhé se tyto dokumenty naskenovaly a následně nahrály do paměti mobilního zařízení. Naskenované dokumenty se pro srovnání převedly také knihovnou Asprise OCR. Výsledky těchto převodů byly zaznamenány do tabulky 5.1, kdy se zjišťoval počet chybně rozpoznaných znaků vzhledem k celkovému počtu znaků v celém dokumentu.

		Aplikace Document Scanner				Knihovna Asprise OCR	
Font	Celkový počet znaků	Celkový počet chybných znaků u fotografie	Úspěšnost rozpoznání [%]	Celkový počet chybných znaků u naskenování	Úspěšnost rozpoznání [%]	Celkový počet chybných znaků u naskenování	Úspěšnost rozpoznání [%]
OCR-A	733	12	98,36	17	97,68	73	90,04
OCR-B	733	15	97,95	26	96,45	27	96,32
Trebouchet MS	747	40	94,65	66	91,16	49	93,44
Lucida Console	749	58	92,26	48	93,59	49	93,46
Verdana	737	96	86,97	85	88,47	38	94,84
Comic Sans MS	910	49	94,62	61	93,30	90	90,11
Arial	898	55	93,88	91	89,87	71	92,09
Times New Roman	914	172	81,18	409	55,25	67	92,67

Tabulka 5.1: Úspěšnost rozpoznání znaků jednotlivých fontů

5.8 Zhodnocení

Z tabulky 5.1 je patrné, že nejlepších výsledků bylo dosaženo u fontů OCR-A a OCR-B. Tyto fonty byly vytvořeny přímo pro tuto technologii a úspěšnost rozpoznání znaků se pohybovala okolo 97%. Naopak jednoznačně nejhoršího výsledku bylo dosaženo u fontu Times New Roman. Tento špatný výsledek je zapříčiněn rozdílnou tloušťkou jednotlivých křivek u každého znaku, kdy použitá knihovna Java OCR měla tendenci tyto znaky rozdělovat na několik dalších znaků. Často se tedy stávalo, že např. znak "D" přeložila knihovna jako dvě závorky "[)" apod.

Co se týče ostatních fontů, nejčastějšími chybami byly záměny malých a velkých písmen, pro jejichž zápis se používá téměř shodné podoby, jako jsou "c, C, o, O, s, S, u, U, v, V, w, W, x, X, z, Z". Kdybychom tyto záměny, které se v textu opakovaly několikrát, nebrali jako chyby, byla by úspěšnost rozpoznávání ještě o něco vyšší. Dále byly časté také záměny znaků podobného tvaru, jako jsou "0, O" a "5, S".

U většiny fontů bylo méně chyb při vyfocení dokumentu než při jeho naskenování, což bývá zpravidla opačně. Naskenovaný dokument totiž neobsahuje žádné rušivé elementy, jako jsou stíny, zaoblení textu apod. Tento výsledek však může vyjít na každém mobilním zařízení jinak, a to především kvůli kvalitě a rozlišení fotoaparátu.

Na OCR knihovně závisí úspěch celé aplikace. Použitá knihovna Java OCR se ukázala jako schopná rozpoznat většinu nahraných fontů. I přesto, že je volně dostupná, nezaostávala za placenou Asprise OCR knihovnou. Při rozpoznávání diakritických a interpunkčních znamének a dalších speciálních znaků dokonce tuto knihovnu předčila. Nejméně chyb zaznamenala při rozpoznávání čísel, naopak největší problém vznikal při vytváření training dat s háčky a čárkami, kdy měla tato knihovna často tendenci dělit znaky na dva další.

6 Závěr

V teoretické části této práce jsem se zabýval operačním systémem Android a technologií OCR. U Androidu jsem popsal jeho historii, architekturu a co přinesly jednotlivé verze. Dále jsem popsal Android z programátorského hlediska, tedy co všechno je potřeba splnit a znát při vývoji mobilních aplikací na tuto platformu. Nakonec jsem vytvořil návod na jednoduchou aplikaci. U technologie OCR jsem uvedl její historii a princip rozpoznávání tištěných textů. Dále jsem uvedl také základní metody používáné při rozpoznávání textu a metody zlepšující kvalitu vstupních obrázků. V závěru teoretické části jsem se zabýval OCR knihovnami pro programovací jazyk Java.

V praktické části práce jsem navázal na znalosti získané v teoretické části. Nejprve jsem zmínil obecné problémy vznikající při vývoji aplikací. Následně popisuji vytvořenou mobilní aplikaci implementující prvky z teoretické části. Uvádím zde použíté metody pro zlepšení kvality vstupního obrázku a metody OCR knihovny vedoucí k rozpoznání tištěného textu. Poté jsem tuto aplikaci otestoval pro všechny nahrané fonty a porovnal s placenou knihovnou Asprise OCR. Nakonec jsem zhodnotil výsledky převodu a použitou OCR knihovnu.

U většiny nahraných fontů dosáhla aplikace dobrých výsledků rozpoznání srovnatelných s placenou knihovnou Asprise OCR. Použitá knihovna Java OCR se však nemůže rovnat dnešním nejlepším placeným OCR knihovnám, jako je např. Abby Fine Reader Engine. Nenabízí totiž takovou kvalitu převodu ani tolik možností při rozpoznávání tištěných textů. Při dalším vývoji bych tuto knihovnu určitě nahradil jinou schopnější knihovnou. Pokud by to již knihovna neumožňovala, přidal bych metody pro rozpoznávání fontů a stylů písma. Dále bych implementoval metody pro narovnání ohnutých či zakřivených textů, které zhoršovaly rozpoznávání znaků, a další metody pro možnost rozpoznávání barevných textů v obrázku. To vše však velice ovlivní rychlost převodu.

Před touto prací jsem zvládl vyvíjet základní jednoduché aplikace na Android, avšak tato práce mé znalosti dále vylepšila. Naučil jsem se další techniky vývoje, především používat hardwarové kamery mobilního zařízení a reagovat na různé doteky a pohyby po displeji. Dále jsem pochopil pricnip rozpoznávání tištěných textů a poznal několik metod pro práci s obrázky.

Použitá literatura

[1] MURPHY, Mark L. *Android 2: průvodce programováním mobilních aplikací*. Vyd. 1. Brno: Computer Press, 2011, 375 s. ISBN 978-80-251-3194-7.

[2] Android. *Openhandsetalliance.com* [online]. [cit. 2012-12-15]. Dostupné z: http://www.openhandsetalliance.com/android_overview.html

[3] Android, the world's most popular mobile platform. *Developer.android.com* [online]. [cit. 2012-12-15]. Dostupné z: http://developer.android.com/about/index.html

[4] Android: Historie. *Osmz.wikidot.com* [online]. 2010 [cit. 2012-12-15]. Dostupné z: http://osmz.wikidot.com/android

[5] Google Android - velký výlet do historie. *Cnews.cz* [online]. 2011 [cit. 2012-12-15]. Dostupné z: http://www.cnews.cz/google-android-velky-vylet-do-historie

[6] App Framework. *Developer.android.com* [online]. [cit. 2012-12-15]. Dostupné z: http://developer.android.com/about/versions/index.html

[7] Platform Versions. *Developer.android.com* [online]. [cit. 2012-12-15]. Dostupné z: http://developer.android.com/about/dashboards/index.html

[8] Developer Tools. *Developer.android.com* [online]. [cit. 2012-12-15]. Dostupné z: http://developer.android.com/tools/index.html

[9] Activity. *Developer.android.com* [online]. [cit. 2012-12-15]. Dostupné z: http://developer.android.com/reference/android/app/Activity.html

[10] KRUMNIKL, Michal. *Operating Systems* [online]. Ostrava, 2012 [cit. 2012-12-20]. Dostupné z: http://tamz2.mrl.cz/download/TAMZ2-2012-0.pdf

[11] CHALOUPKA, Milan. *Rozpoznávání textu a technologie OCR* [online]. Ostrava, 2010 [cit. 2013-03-28]. Dostupné z: http://dspace.vsb.cz/handle/10084/78492. Diplomová práce. Vysoká škola báňská - Technická univerzita ostrava.

[12] ŠLAJCHRT, Jan. *Technologie převodu textu z tištěné podoby do elektronické* [online]. Praha, 2009 [cit. 2013-03-28]. Dostupné z: info.sks.cz/www/zavprace/soubory/68400.pdf. Bakalářská práce. Vysoká škola ekonomická Praha.

[13] ZÍKOVÁ, Naděžda. *Prostorová variabilita ročních chodů atmosférických srážek* [online]. Praha, 2009 [cit. 2013-03-29]. Dostupné z: http://kmop.mff.cuni.cz/docs/users/students/zikova/DP_Zikova.pdf. Diplomová práce. Univerzita Karlova v Praze.

Seznam příloh

Příloha.A: Testovací dokument pro font OCR-A I

Příloha.B: Testovací dokument pro font OCR-B II

Příloha.C: Testovací dokument pro font Trebouchet MS III

Příloha.D: Testovací dokument pro font Lucida Console IV

Příloha.E: Testovací dokument pro font Verdana V

Příloha.F: Testovací dokument pro font Comic Sans MS VI

Příloha.G: Testovací dokument pro font Arial VII

Příloha.H: Testovací dokument pro font Times New Roman VIII

Součástí BP je CD.

Adresářová struktura přiloženého CD:

Název složky	**Popis**
Aplikace	Instalační soubor aplikace
Vypracování	Vypracování BP v elektronické podobě
Zdrojové_kódy	Zdrojové kódy aplikace

Font OCR-A

!"#$%&'()*+,-./0123456789:;<=>?@

ABCDEFGHIJKLMNOPQRSTUVWXYZ
[\]^_`abcdefghijklmnopqrstuvwxyz{|}~

This is the test document for my new application called Document Scanner. This font is called OCR-A, size is 16 and style is normal. Take picture of this document and evaluate it by Document Scanner application. For questions send email on libor.stebel@gmail.com or call 728707263. Thank you!

This is another test text that doesn't make sense. It should only use another "symbols". Start with dollar 15$, hashtag #ff00ee, percentage 36%, ampersand &amp, (parentheses), operators 2*4+3-6/2=8?, colon:semicolon; <arrows>, question mark?[brackets],backslash\, arrow up^, underscore_, {brackets}, pipeline | and tilde~.

0123456789876543210110011100011001O

03.06.2013 11.12.2093 5.4.1980

+420723758263 596458423 0045687954321792

Font OCR-B

!"#$%&'()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ
[\]^_`abcdefghijklmnopqrstuvwxyz{|}~

This is the test document for my new application called Document Scanner. This font is called OCR-B, size is 16 and style is normal. Take picture of this document and evaluate it by Document Scanner application. For questions send email on libor.stebel@gmail.com or call 728707263. Thank you!

This is another test text that doesn't make sense. It should only use another "symbols". Start with dollar 15$, hashtag #ff00ee, percentage 36%, ampersand &amp, (parentheses), operators 2*4+3-6/2=8?, colon:semicolon; <arrows>, question mark?[brackets],backslash\, arrow up^, underscore_, {brackets}, pipeline | and tilde~.

0123456789876543210110011100011001O

03.06.2013 11.12.2093 5.4.1980

+420723758263 596458423 004568795432179Z

Font Trebouchet MS

!"#$%&'()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ
[\]^_`abcdefghijklmnopqrstuvwxyz{|}~

This is the test document for my new application called Document Scanner. This font is called Trebouchet MS, size is 16 and style is normal. Take picture of this document and evaluate it by Document Scanner application. For questions send email on libor.stebel@gmail.com or call 728707263. Thank you!

This is another test text that doesn't make sense. It should only use another "symbols". Start with dollar 15$, hashtag #ff00ee, percentage 36%, ampersand &amp, (parentheses), operators 2*4+3-6/2=8?, colon:semicolon; <arrows>, question mark?[brackets],backslash\, arrow up^, underscore_, {brackets}, pipeline | and tilde~.

01234567898765432101100111000110010

03.06.2013 11.12.2093 5.4.1980

+420723758263 596458423 0045687954321792

Font Lucida Console

```
!"#$%&'()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ
[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
```

This is the test document for my new application called Document Scanner. This font is called Lucida Console, size is 16 and style is normal. Take picture of this document and evaluate it by Document Scanner application. For questions send email on libor.stebel@gmail.com or call 728707263. Thank you!

This is another test text that doesn't make sense. It should only use another "symbols". Start with dollar 15$, hashtag #ff00ee, percentage 36%, ampersand &amp, (parentheses), operators 2*4+3-6/2=8?, colon:semicolon; <arrows>, question mark?[brackets],backslash\, arrow up^, underscore_, {brackets}, pipeline | and tilde~.

0123456789876543210110011100011001003.06.2013 11.12.2093 5.4.1980

+420723758263 596458423 004568795432179 2

Font Verdana

!"#$%&'()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ
[\]^_`abcdefghijklmnopqrstuvwxyz{|}~

This is the test document for my new application called Document Scanner. This font is called Verdana, size is 16 and style is normal. Take picture of this document and evaluate it by Document Scanner application. For questions send email on libor.stebel@gmail.com or call 728707263. Thank you!

This is another test text that doesn't make sense. It should only use another "symbols". Start with dollar 15$, hashtag #ff00ee, percentage 36%, ampersand &amp, (parentheses), operators 2*4+3-6/2=8?, colon:semicolon; <arrows>, question mark?[brackets],backslash\, arrow up^, underscore_, {brackets}, pipeline | and tilde~.

0123456789876543210110011100011001003.06.2013 11.12.2093 5.4.1980

+420723758263 596458423 004568795432179 2

Font Comic Sans MS

!"#$%&'()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ
[\]^_`abcdefghijklmnopqrstuvwxyz{|}~

This is the test document for my new application called Document Scanner. This font is called Comic Sans MS, size is 16 and style is normal. Take picture of this document and evaluate it by Document Scanner application. For questions send email on libor.stebel@gmail.com or call 728707263. Thank you!

This is another test text that doesn't make sense. It should only use another "symbols". Start with dollar 15$, hashtag #ff00ee, percentage 36%, ampersand &amp, (parentheses), operators 2*4+3-6/2=8?, colon:semicolon; <arrows>, question mark?[brackets],backslash\, arrow up^, underscore_, {brackets}, pipeline | and tilde~.

0123456789876543210110011100011001O

03.06.2013 11.12.2093 5.4.1980

+420723758263 596458423 004568795432l792

Tento font podporuje některé háčky a čárky. Stebel©

á â Á Č č Ď ď é É Ě ě í Í ý Ň ň ó Ó
Ř ř Š š Ť ť Ú ú Ů ů Ž ž ™ © ® ¶

Žáci Ámos Čedič Ďeď® Řeřicha Šašek Ťapkâ™ Úplný Ódaó Ítý Ůleť žvýkačka český znaků nemůžeš šířit tůňka ¶

Font Arial

!"#$%&'()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ
[\]^_`abcdefghijklmnopqrstuvwxyz{|}~

This is the test document for my new application called Document Scanner. This font is called Arial, size is 16 and style is normal. Take picture of this document and evaluate it by Document Scanner application. For questions send email on libor.stebel@gmail.com or call 728707263. Thank you!

This is another test text that doesn't make sense. It should only use another "symbols". Start with dollar 15$, hashtag #ff00ee, percentage 36%, ampersand &amp, (parentheses), operators 2*4+3-6/2=8?, colon:semicolon; <arrows>, question mark?[brackets],backslash\, arrow up^, underscore_, {brackets}, pipeline | and tilde~.

0123456789876543210110011100011001 0

03.06.2013 11.12.2093 5.4.1980

+420723758263 596458423 004568795432179 2

Tento font podporuje některé háčky a čárky. Stebel©

á â Á Č č Ď ď é É Ě ě í Í ý Ň ň ó Ó
Ř ř Š š Ť ť Ú ú Ů ů Ž ž ™ © ® ¶

Žáci Ámos Čedič Ďeď® Řeřicha Šašek Ťapkâ™ Úplný Ódaó Ítý Ůleť žvýkačka český znaků nemůžeš šířit tůňka ¶

Font Times New Roman

!"#$%&'()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ
[\]^_`abcdefghijklmnopqrstuvwxyz{|}~

This is the test document for my new application called Document Scanner. This font is called Times New Roman, size is 16 and style is normal. Take picture of this document and evaluate it by Document Scanner application. For questions send email on libor.stebel@gmail.com or call 728707263. Thank you!

This is another test text that doesn't make sense. It should only use another "symbols". Start with dollar 15$, hashtag #ff00ee, percentage 36%, ampersand &amp, (parentheses), operators 2*4+3-6/2=8?, colon:semicolon; <arrows>, question mark?[brackets],backslash\, arrow up^, underscore_, {brackets}, pipeline | and tilde~.

01234567898765432101100111000110010

03.06.2013 11.12.2093 5.4.1980

+420723758263 596458423 0045687954321792

Tento font podporuje některé háčky a čárky. Stebel©

á â Á Č č Ď ď é É Ě ě í Í ý Ň ň ó Ó
Ř ř Š š Ť ť Ú ú Ů ů Ž ž ™ © ® ¶

Žáci Ámos Čedič Ďeď® Řeřicha Šašek Ťapkâ™ Úplný Ódaó Ítý Ůleť žvýkačka český znaků nemůžeš šířit tůňka ¶